# MQL4 初级培训教程

作者：老易
QQ：921795
2011 年 5 月 25 日

目录

# 第一章　开始使用 MT4

## 1.1 前言

当变幻莫测的外汇市场、24 小时不间断的交易、品种繁多的货币对同时展现在你的眼前时，你一定有手忙脚乱无所适从的感觉。自从实现了互联网外汇交易，我们倍感外汇交易的繁重与繁琐，于是 EA（英文 Expert Advisors 缩写，称专家顾问，或智能交易系统）就应运而生了。

大多数外汇交易商提供 MT4 平台，大多数外汇交易者开始关注甚至迷恋 MT4 平台上的 EA，网上出现了很多的免费 EA 甚至收费的 EA。不幸的是，在这里我要下个结论，那就是大多数 EA 都是垃圾绝不是“圣杯”，不管是免费的还是收费的，真正的圣杯只能在你自己手中诞生。

纵观历年国际上 EA 大赛，还没有出现一位连续获胜的选手。或许我们可以暂时认为连续稳定获利的交易系统是存在的，但是连续稳定获利的 EA 是否存在则有待观察证实。电脑和人脑相比目前还存在难以逾越的障碍，我们期盼并等待着众多的专家学者能制造出真正的人工智能交易系统。

然而，在所谓真正的人工智能交易系统问世之前，作为普通的炒汇者不能闲着，我们需要积极的做些什么来得到自己的圣杯。

有一点可以肯定，我们必须在正确的市场观和深刻认识市场的基础上去构建适合自己的方法，制定市场适应能力较强的策略，保证系统能够动态地以最贴近市场的方式运行，再通过整理交易过程的逻辑规则，按照 MQL4 语言规范编出适合电脑自动交易的程序，就可以阶段性的实现实现稳定盈利。

EA 的最大用途就在于把正确的交易逻辑设计定量化、程序化，从而创建一套市场适应能力较强的策略。切记 EA 只是你交易行为的一部分，切忌 EA 左右你的交易行为。你必须全程参与到整个交易过程中，如果你过分迷恋 EA，那么 EA 就只能是个传说。

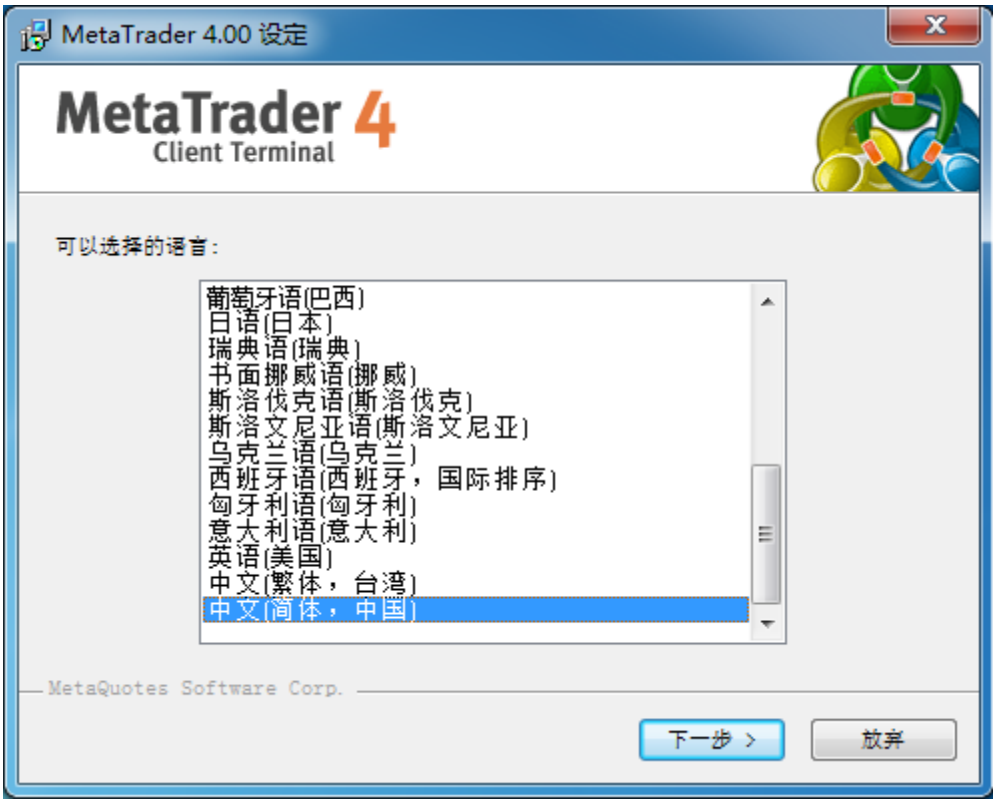
本书将从搭建交易平台、了解自动交易编程、学习编程等方面分章节描述，并贯穿若干个 EA 实例程序，按照构思策略、逻辑分析、编制代码、历史数据测试、模拟操盘的顺序，深度全面地诠释 EA 的诞生过程，同时提供了 MQL4 常用指令集、外汇常用技术指标解释等内容。

笔者既不属于消息派也不属于技术派，更不是二合一派。外汇交易是“零和博弈”，我更偏向从数学统计论的角度来思考外汇，理性的参与博弈。

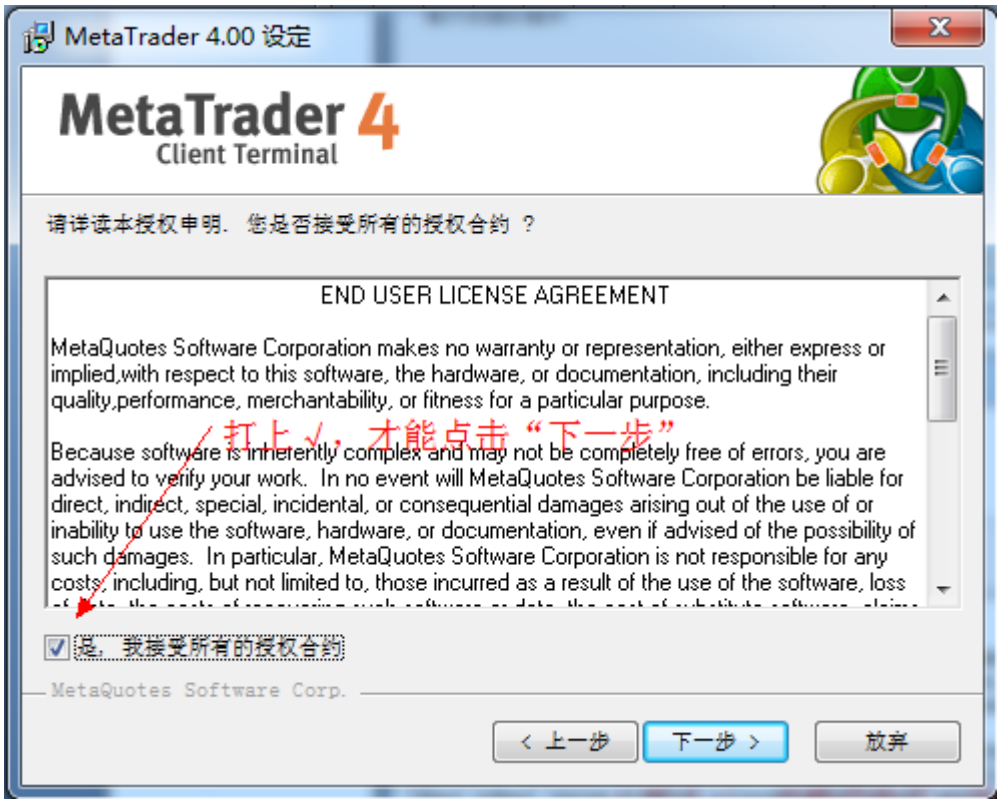
理解 EA，编制 EA，使用 EA，从现在开始。

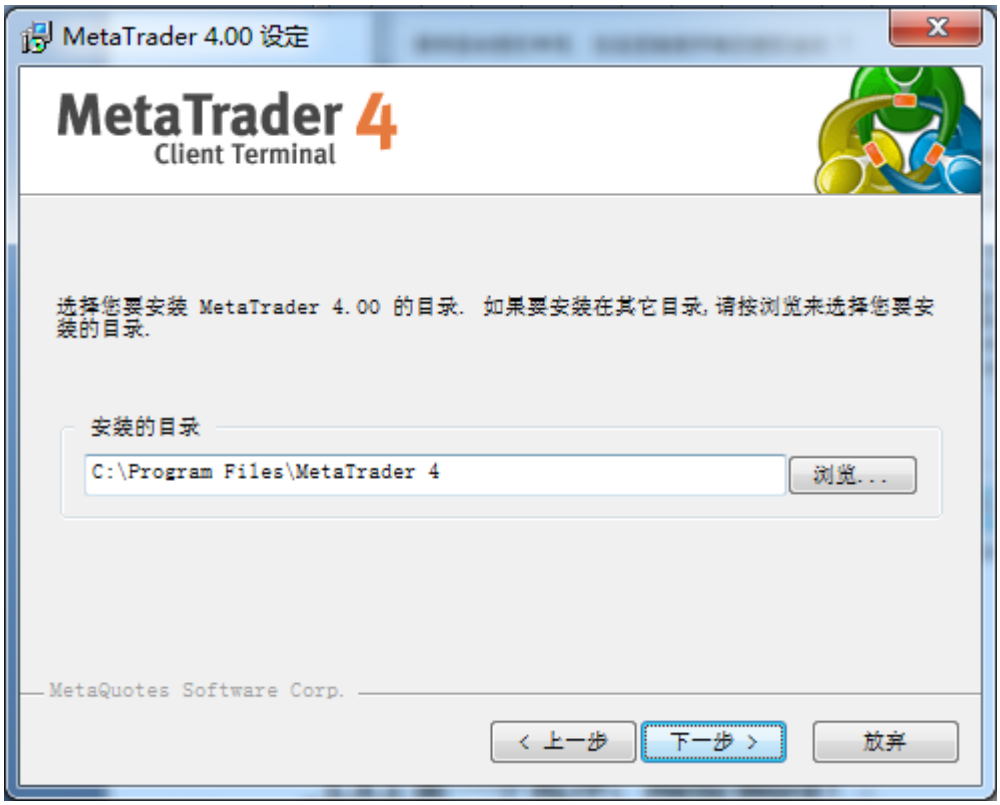
## 1.2 MT4 下载与安装

在交易商指定的网站上下载安装程序，并运行：

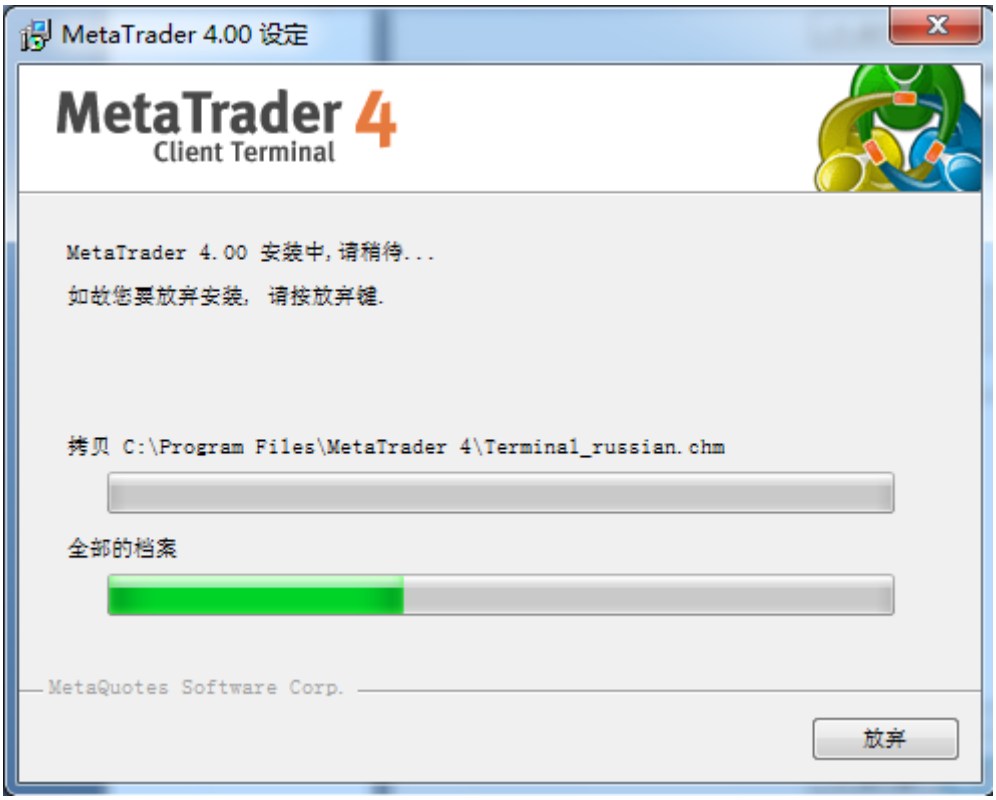

MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
可以选择的语言：
葡萄牙语(巴西)
日语(日本)
瑞典语(瑞典)
书面挪威语(挪威)
斯洛伐克语(斯洛伐克)
斯洛文尼亚语(斯洛文尼亚)
乌克兰语(乌克兰)
西班牙语(西班牙，国际排序)
匈牙利语(匈牙利)
意大利语(意大利)
英语(美国)
中文(繁体，台湾)
中文(简体，中国)
MetaQuotes Software Corp.
下一步 >
放弃

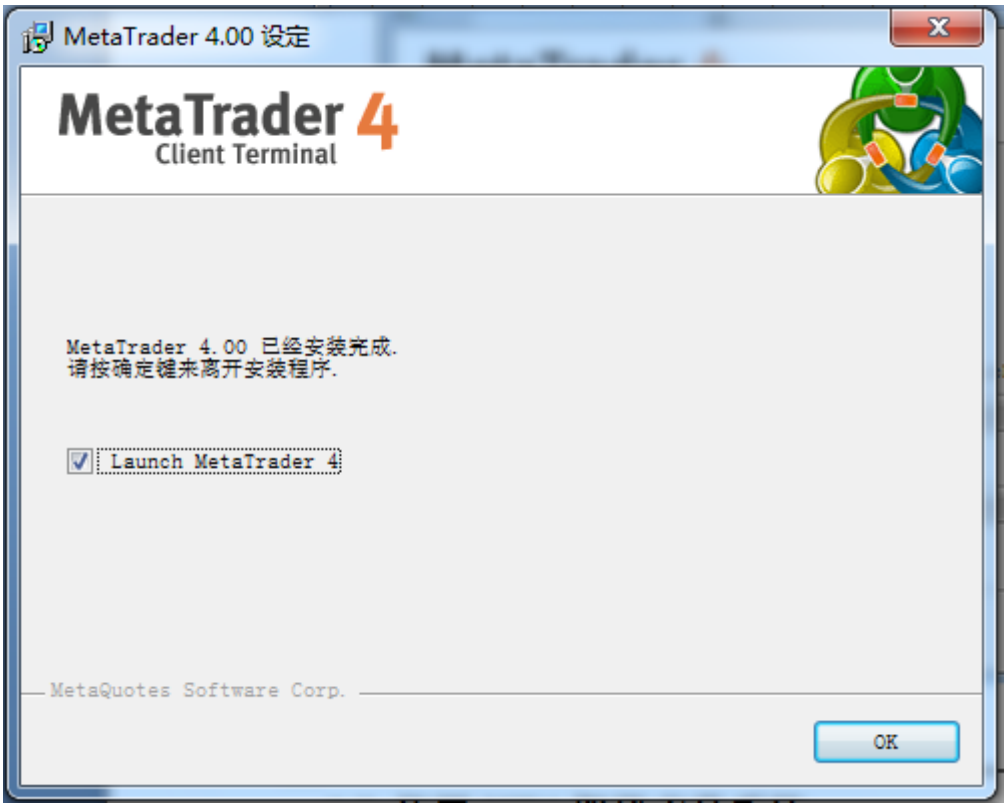

MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
欢迎您使用 MetaTrader 4.00 安装程序. 此安装程序将会安装 MetaTrader 4.00 在您的计算机上.
建议您在安装前先关闭其它正在执行应用软件. 在您安装的过程中，可以按放弃键来终止安装. 请按下一步来继续安装.
敬告：这个程序受著作权法及国际公约之保护.
未经授权而散布或复制程序的全部或部分，将导致严厉的民事及刑事处分，且将被依法提起最大范围的追诉.
MetaQuotes Software Corp.
< 上一步
下一步 >
放弃


MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
请详读本授权申明. 您是否接受所有的授权合约 ?
END USER LICENSE AGREEMENT
MetaQuotes Software Corporation makes no warranty or representation, either express or implied,with respect to this software, the hardware, or documentation, including their quality,performance, merchantability, or fitness for a particular purpose.
打上√，才能点击“下一步”
Because software is inherently complex and may not be completely free of errors, you are advised to verify your work. In no event will MetaQuotes Software Corporation be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of or inability to use the software, hardware, or documentation, even if advised of the possibility of such damages. In particular, MetaQuotes Software Corporation is not responsible for any costs, including, but not limited to, those incurred as a result of the use of the software, loss
是, 我接受所有的授权合约
MetaQuotes Software Corp.
< 上一步
下一步 >
放弃


MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
选择您要安装 MetaTrader 4.00 的目录. 如果要安装在其它目录,请按浏览来选择您要安装的目录.
安装的目录
C:\Program Files\MetaTrader 4
浏览...
MetaQuotes Software Corp.
< 上一步
下一步 >
放弃


MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
选择程序群组:
MetaTrader 4
360杀毒
Accessories
Administrative Tools
Games
Lenovo
Maintenance
Microsoft Office
Second Life Viewer 2
Startup
Tablet PC
Windows Live
WinRAR
搜狗拼音输入法
MetaQuotes Software Corp.
< 上一步
下一步 >
放弃


MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
MetaTrader 4.00 安装中,请稍待...
如欲您要放弃安装, 请按放弃键.
拷贝 C:\Program Files\MetaTrader 4\Terminal_russian.chm
全部的档案
MetaQuotes Software Corp.
放弃


MetaTrader 4.00 设定
MetaTrader 4
Client Terminal
MetaTrader 4.00 已经安装完成.
请按确定键来离开安装程序.
Launch MetaTrader 4
MetaQuotes Software Corp.
OK

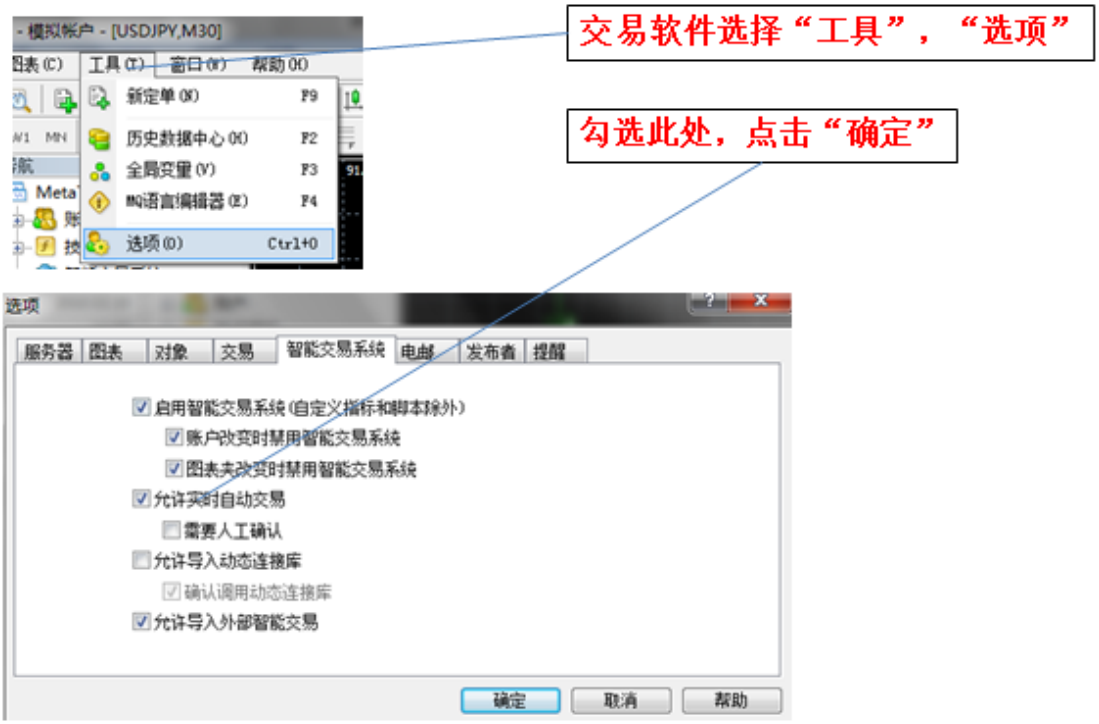
## 1.3 熟悉软件环境


报价窗口
导航窗口
终端窗口
图表窗口
MetaTrader - [EURUSD,H4]
MetaTrader
MetaQuotes Software Corp.
What Is Automated Trading?
Mobile Trading - It's Easy!
Welcome!

# 1.4 使用 MT4 智能交易系统

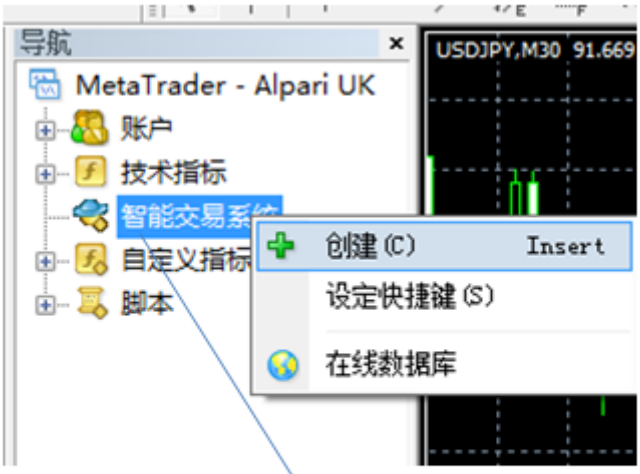
## 1.4.1 智能交易系统设置

MT4 安装运行后，智能交易是被禁止的，需要设置“允许”智能交易。



## 1.4.2 第一个程序： Hello Word！

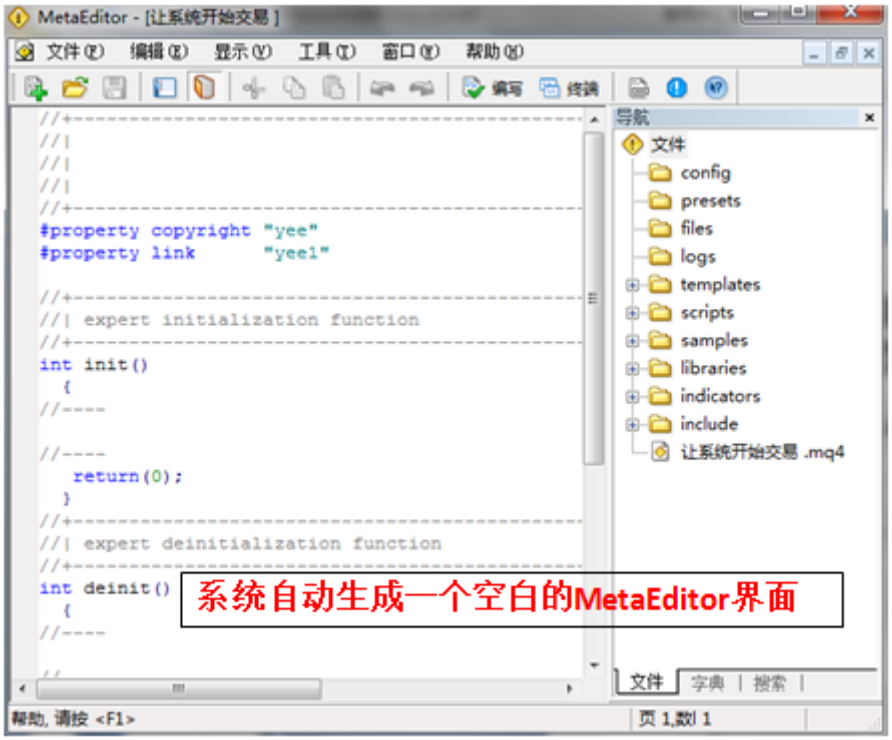
### 1.4.2.1创建一个空白的程序



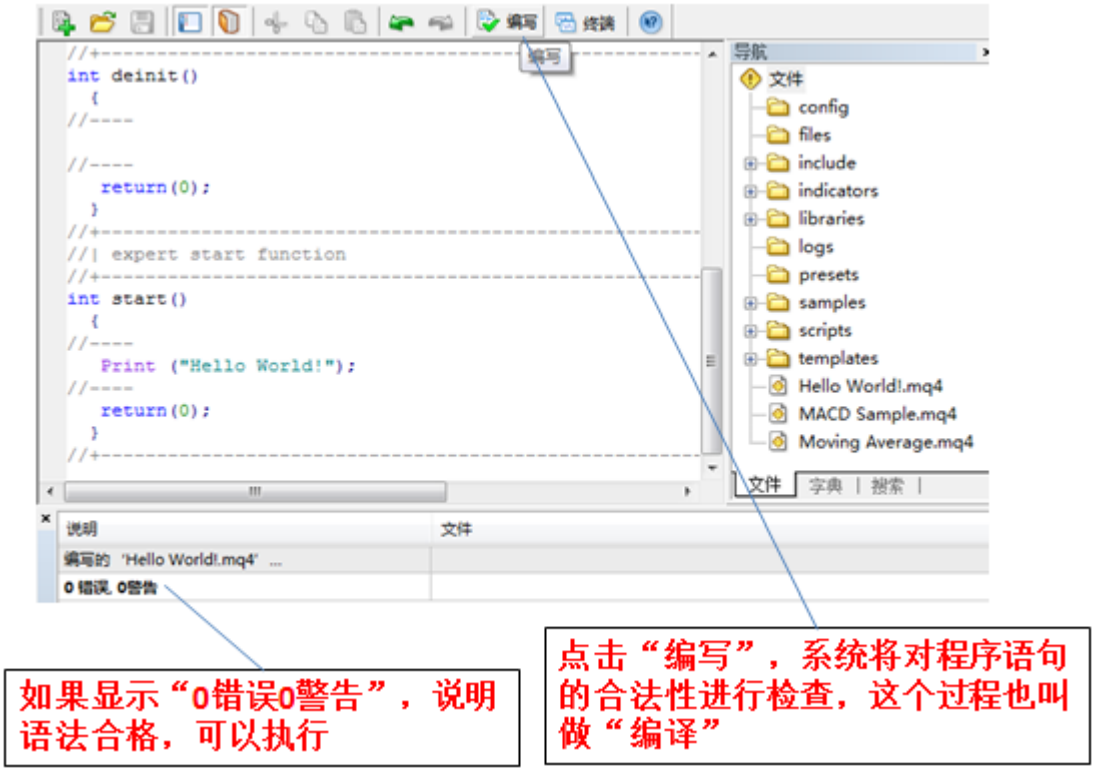

点击右键，选择“创建”


点击“下一步”

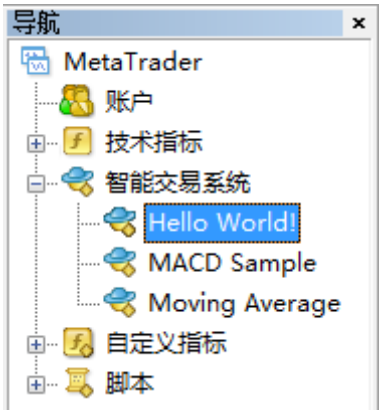

智能交易创作
智能交易的常规属性
请指定智能交易的常规属性
名称: 让系统开始交易
作者: yee
字行: yee1
参数:
名称
类型
无效值
添加
删除(D)
< 上一步(B)
完成
取消
填写一个容易识别的名称
最后点击“完成”


MetaEditor - [让系统开始交易 ]
文件(F) 编辑(E) 显示(V) 工具(T) 窗口(W) 帮助(H)
编写
终端
导航
文件
config
presets
files
logs
templates
scripts
samples
libraries
indicators
include
让系统开始交易 .mq4
#property copyright "yee"
#property link "yee1"
int init()
int deinit()
系统自动生成一个空白的MetaEditor界面
文件
字典
搜索
帮助, 请按 <F1>
页 1,数 1

### 1.4.2.2写一个程序

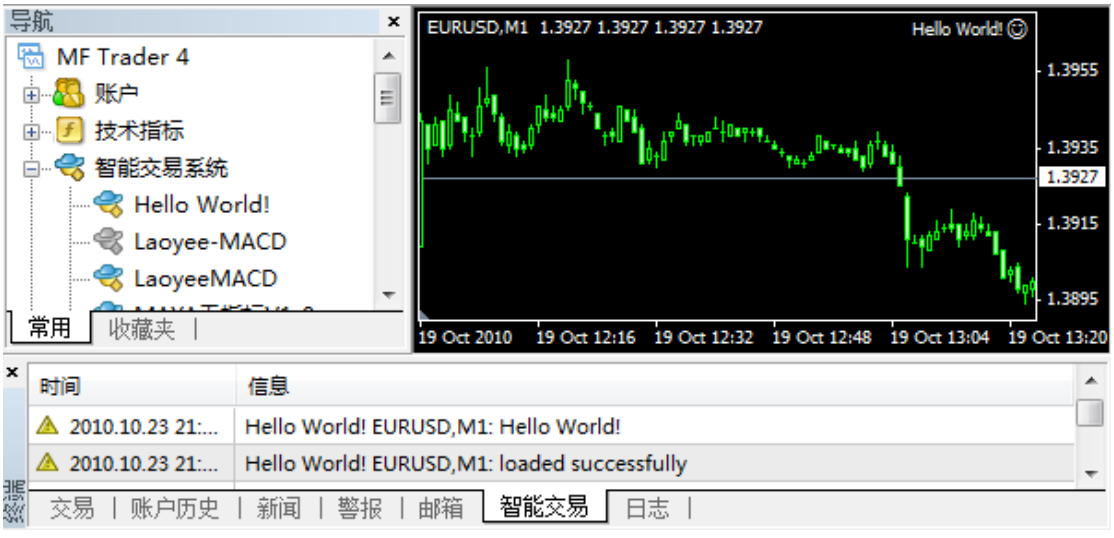




回到 MT4 终端，打开智能交易系统可以看到刚才的程序已经准备好了：




鼠标双击这个名字，新编写好的程序就调入到图表窗口，右上角有程序名和一个笑脸，笑脸表示系统允许执行自动交易：


我们在终端窗口的智能交易标签里能看到程序运行的结果：

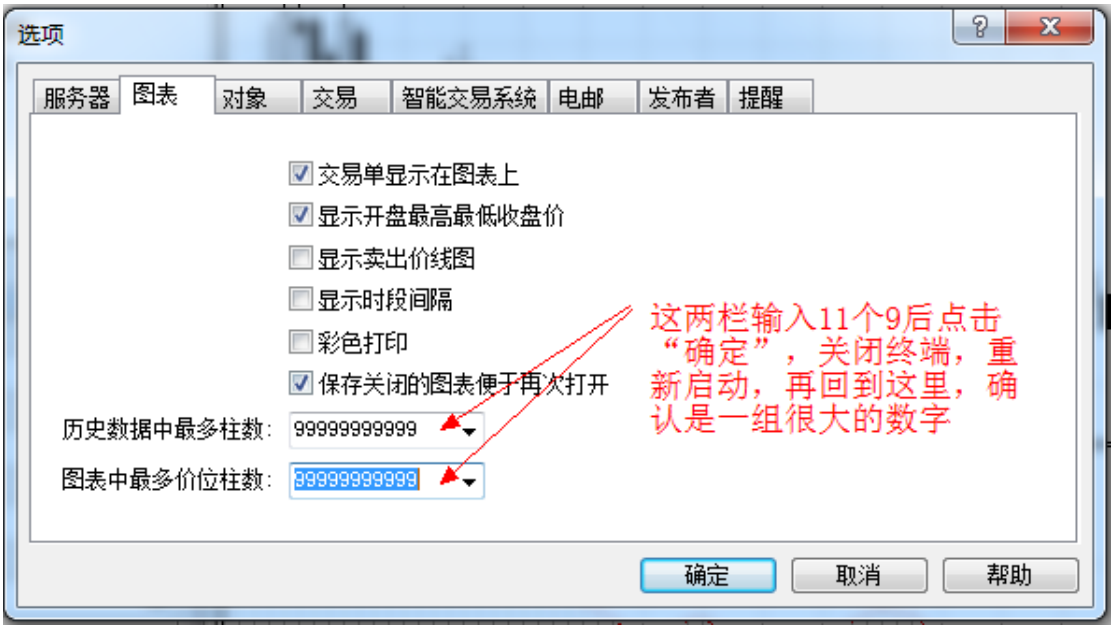



## 1.4.3 准备 10 年的历史数据

一个编制好的 EA 需要历史数据回测验证。目前从 MT4 平台上可以下载从 1999 年 10 月以来的所有品种的数据（包括外汇、黄金、期货、股票指数等等）。

下载 10 年的数据按照以下方法即可。

第一步，设置终端图表显示参数。打开终端“工具”->“选项”，点击“图表”标签。





第二步，下载历史数据。打开终端“工具”->“历史数据中心”。


这一步下载了选定货币对的从 1999 年 10 月 1 日以来的所有 1 分钟数据。其他时间周期的数据都会根据这个 M1 数据自动生成，不过，你还需要双击每个时间周期，让所有的周期都显示成彩色的。这说明所有时间周期的数据都将被终端调用。

由于网络或者服务器的原因，你可能需要反复点击货币对和下载按钮，直到 1999 年数据显示为止。

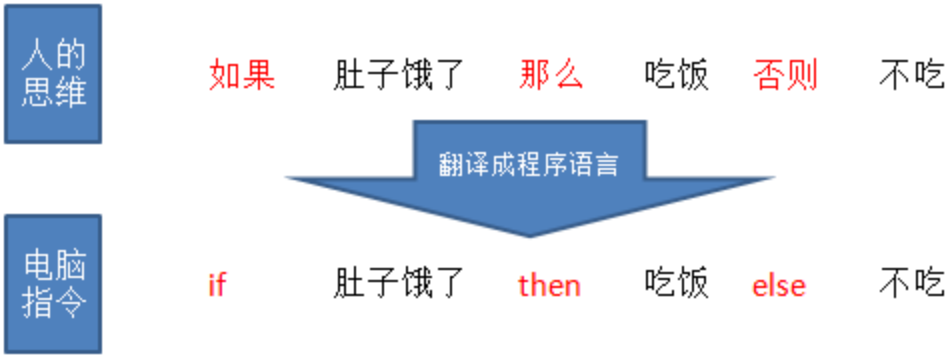
其他品种以此类推。

# 第二章　MQL4 语言

## 2.1 预备知识

在学习 MQL4 语言前，首先要打消自己的顾虑，不要被网上流传的“写 MQL4 程序必须具备 C 语言基础”给吓到，大多数人学不会编程就是自己把自己劝退的。

当然，学习计算机语言要求你必须有很好的逻辑思维能力。我们可以通过下面的内容来理解计算机的逻辑。



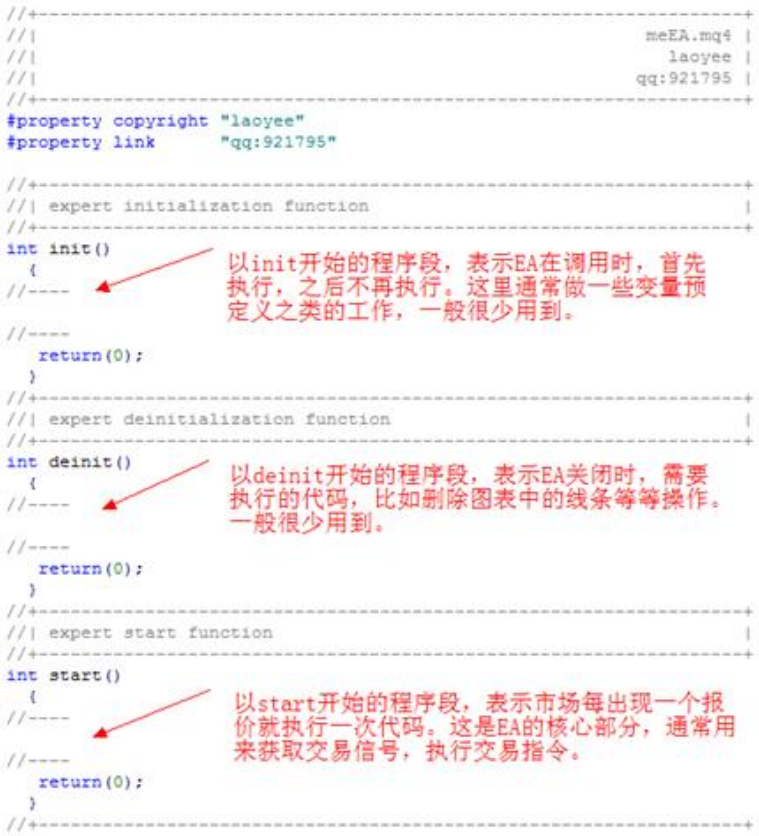

所有的计算机语言都包含两个语句，一个是条件（if）语句一个是循环（for）语句。

If 语句顾名思义，满足条件就执行，否则就跳过。

for 语句顾名思义，就是在一定条件下反复执行规定的指令，直到条件不满足。

### 2.1.1 EA 框架

标准的 EA 由 5 个部分组成，分别是变量预定义、EA 初始化程序、EA 结束程序、EA 执行程序和自定义变量，如下图：




在 int init()的前面是 EA 变量定义区，当 EA 调入到图标时，需要做预设置的参数都在这里定义。

在 int start()后面都是自定义函数区间。

## 2.1.2 指标框架

与 EA 框架一样。区别如下：

1、 变量预定义部分的首行必须定义该指标显示的位置，主图和副图。

2、 EA 初始化程序部分用来定义显示元素的属性。

3、 EA 执行程序部分用来计算指标值。

如下图：

```
//+------------------------------------------------------------------+
//|                                                  myIndicator.mq4 |
//|                                                           laoyee |
//|                                                    qq:921795 |
//+------------------------------------------------------------------+
#property copyright "laoyee"
#property link      "qq:921795"

#property indicator_chart_window
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
//---- indicators
//----
   return(0);
  }
//+------------------------------------------------------------------+
//| Custom indicator deinitialization function                       |
//+------------------------------------------------------------------+
int deinit()
  {
//----

//----
   return(0);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int start()
  {
   int    counted_bars=IndicatorCounted();
//----

//----
   return(0);
  }
```

预定义图表在主图显示。这部分通常用来预定义指标在图表中显示线条、箭头等元素

与EA框架类似，不同的是在这个部分定义线条、箭头的类型、颜色、大小等参数

与EA框架一样

计算线条的节点坐标，或者箭头坐标，在图表中就能自动显示了

## 2.1.3 坐标系

自动交易的执行是需要准确定位的，因此就必须建立起明晰坐标系的概念。

图上的横坐标既可以是市场时间也可以是蜡烛序号，在编程的时候通常使用蜡烛序号。

坐标系实际上是三维的，即时间、价格、开仓量，第三维开仓量通常在风险控制策略中考虑，比如出现亏损加大开仓量，或者亏损 20%平仓等等，因此我们编写程序重点就在时间和价格这二维空间中。


建立起正确的坐标系概念是编程的基础，因为你即将对技术指标进行分析，计算开仓平仓信号，甚至在图表上画线做标记。

## 2.2 内置变量与函数

MQL4 提供了大量的内置变量与函数，用来取值计算。目前网上有许多的手册，但都是翻译机器从原版英文手册自动翻译的，可读性极差。作者通过积累大量的经验加上自己的理解，精选了部分常用的、实用的内容重新说明如下。

### 2.2.1 整数相除的方法

在 MQL4 的语法中有“+ - * /”四则运算，当你直接用“1/3”的时候，会返回 0。在程序中可以这么来实现的：

```
double i=(1*0.01)/(3*0.01);
```

这时变量 i 才会返回你所要的值：0.33333333。

### 2.2.2 市场函数

我们经常能遇到不同平台报价格式不同、滑点数不同、最小开仓量不同、市场时间不同等等情况。这些数据都能通过市场函数直接获取，这样才能给 EA 带来了较大的适用性。

市场函数调用范例：定义最低价变量 myLow，并获取最低价。

| | |
|---|---|
| MarketInfo(symbol(),MODE_LOW) | 获取当前货币对的最低价 |

所有参数列表如下：

| 常数 | 描述 |
|---|---|
| MODE_LOW | 当日最低价 |
| MODE_HIGH | 价格最高日 |
| MODE_TIME | 最后价格变动时间（服务器显示时间） |
| MODE_BID | 市场最新买入叫价，如果你要卖出则按照这个价格执行 |

| | |
|---|---|
| MODE_ASK | 市场最新卖出叫价，如果你买入则按照这个价格成交 |
| MODE_POINT | 价格最小变动单位，例如 USDJPY 为 0.01，有的平台为 0.001 |
| MODE_DIGITS | 货币交易价格小数点位数，比如 2 位、4 位、5 位 |
| MODE_SPREAD | 买入叫价与卖出叫价的差价，也叫“点差”。为交易商收取的手续费。例如现在需要买入 1 手，那么成交价就是“卖出叫价”，反之则是“买入叫价”，成交后会与市场价格形成一个差价 |
| MODE_STOPLEVEL | 平仓点差。设置止损止赢点时只允许在这张订单价格±平仓点差范围之外。例如 USDJPY 成交价为 91.75，平仓点差为 5，那么止损止盈点设置必须在 91.70~91.80 范围之外 |
| MODE_LOTSIZE | 基本货币的标准手大小，例如：USDJPY 为 100000 美元，GBPUSD 为 100000 英镑，EURUSD 为 100000 欧元 |
| MODE_TICKVALUE | 1 手每点本币的价值，例如 USDJPY 当价格为 91.90 时 1 手每点价值$10.8841，当价格变成 91.88 时 1 手每点价值为$10.8838。UERUSD 恒定为 10 欧元，GBPUSD 恒定为 10 英镑。这个值是交易商用来计算平仓时计算实际货币的依据 |
| MODE_TICKSIZE | 报价最小单位，与上面最小变动单位可能有不同，我不能确定 |
| MODE_SWAPLONG | 多头仓位掉期。应该与结算利息相关，关于掉期的权威解释参见 http://xusun.blog.hexun.com/46091597_d.html |
| MODE_SWAPSHORT | 空头仓位掉期。 |
| MODE_STARTING | 市场开始日期（预留常量），一般为 0 |
| MODE_EXPIRATION | 市场时间周期（预留常量），一般为 0 |
| MODE_TRADEALLOWED | 交易允许货币对数量，所有货币对都为 1 |
| MODE_MINLOT | 最小允许标准手数，一般为 0.01 |
| MODE_LOTSTEP | 改变标准手最小单位，一般为 0.01 |
| MODE_MAXLOT | 最大允许标准手数，一般为 10000 手 |
| MODE_SWAPTYPE | 掉期计算方法. 0 - 点; 1 -基本货币对; 2 - 兴趣; 3 - 货币保证金，一般为 0 |
| MODE_PROFITCALCMODE | 赢利计算模式，0 – Forex（外汇）; 1 – CFD（黄金）; 2 – Futrues（期货） |
| MODE_MARGINCALCMODE | 保证金计算模式，0 - Forex; 1 - CFD; 2 - Futrues; 3 - CFD for indices（黄金指数） |
| MODE_MARGININIT | 对于 1 标准手的初始保证金需求，一般为 0 |
| MODE_MARGINMAINTENANCE | 对于 1 标准手开仓的保证金，一般为 0 |
| MODE_MARGINHEDGED | 对于 1 标准手的护盘保证金，一般为 5000 |
| MODE_MARGINREQUIRED | 对于购买一个标准手开仓的自由保证金 |
| MODE_FREEZELEVEL | 冻结定单水平点。如果执行的价格在冻结水平点范围内，定单将会被注销或关闭，这是交易商设置的参数，一般 |

| | |
|---|---|
| | 为 0 |

## 2.2.3 账户函数

| | |
|---|---|
| AccountBalance( ) | 获取账户余额 |
| AccountCredit( ) | 获取账户信用点数 |
| AccountCompany( ) | 获取交易平台公司名称 |
| AccountCurrency( ) | 获取账户通用货币名称 |
| AccountEquity( ) | 获取账户净值 |
| AccountFreeMargin( ) | 获取账户免费保证金 |
| AccountFreeMarginCheck( string symbol, int cmd, double volume) | 获取当前账户的当前价格上在指定开仓的仓位返回自由保证金，即最大可用保证金，价格变动，该值随着变动。不同货币对、不同价位，自由保证金不同。 |
| AccountFreeMarginMode( ) | 在当前开仓位置的账户上计算免费保证金的模式。计算方式可能采取以下价格值:<br>0 - 浮动 profit/loss 不使用<br>1 - 两个浮动赢利和损失在开仓位置上使用计算自由保证金;<br>2 - 只有赢利值被使用计算，不考虑当前开仓的亏损;<br>3 - 只有亏损值被使用计算, 不考虑当前开仓的亏损。 |
| AccountLeverage( ) | 获取账户杠杆比率 |
| AccountMargin( ) | 获取账户被占用的保证金总和 |
| AccountName( ) | 获取账户名称 |
| AccountNumber( ) | 获取账户账号 |
| AccountProfit( ) | 获取账户利润 |
| AccountServer( ) | 获取账户所在服务器名称 |
| AccountStopoutLevel( ) | 获取账户停止水平 |
| AccountStopoutMode( ) | 对于停止水平返回的的运算方式。运算方式值如下:<br>0 - 计算保证金和净值之间的百分比;<br>1 - 比较自由保证金水平和绝对值。 |

## 2.2.4 市场变量

| | |
|---|---|
| Close[i] | 获取第 i 个蜡烛的收盘价，如果 i=0，就是获取当前价 |
| High[i] | 获取第 i 个蜡烛的最高价 |
| Low[i] | 获取第 i 个蜡烛的最低价 |
| Open[i] | 获取第 i 个蜡烛的开盘价 |
| Time[0] | 获取第 i 个蜡烛的时间，这个值是用秒来计算的 |
| Volume[0] | 获取第 i 个蜡烛的成交量 |

## 2.2.5 时间函数

MQL4 内置时间函数数值的最小读取单位是以每个新价格（tick）为基础。如果没有新价格出现，则时间数值不能获取。

Time[0]和 TimeCurrent()的数据类型为 datetime，返回从 1970 年 1 月 1 日 0 点开始至今累计的“秒”数，Time[0]返回当前蜡烛时间，TimeCurrent()返回当前新价格（tick）时间。

| | |
|---|---|
| int Day( ) | 返回当前服务器的日，如 14，表示 14 日 |
| int DayOfWeek( ) | 返回当前服务器的星期，如 4，表示星期 4 |
| int DayOfYear( ) | 返回当前服务器的年，如 2010，表示 2010 年 |
| int Hour( ) | 返回当前服务器的时，如 10，表示 10 点 |
| int Minute( ) | 返回当前服务器的分，如 15，表示 15 分 |
| int Month( ) | 返回当前服务器的月，如 10，表示 10 月 |
| int Seconds( ) | 返回当前服务器的秒，如 34，表示 34 秒 |
| datetime TimeCurrent( ) | 返回当前服务器最新价格的秒，该数值表示从 1970 年 1 月 1 日至今累计秒。 |
| int TimeDay( datetime date) | 返回日期类型参数中的日 |
| int TimeDayOfWeek( datetime date) | 返回日期类型参数中当周的天数，如 4，表示当周的第 4 天 |
| int TimeDayOfYear( datetime date) | 返回日期类型参数中当年的天数，如 287，表示当年的第 287 天 |
| int TimeHour( datetime time) | 返回日期类型参数中当天的小时数，如 5，表示当天的第 5 个小时 |
| datetime TimeLocal( ) | 返回本地计算机当前时间，以秒为单位 |
| int TimeMinute( datetime time) | 返回日期类型参数中的分钟数，如 17，表示第 17 分钟 |
| int TimeMonth( datetime time) | 返回日期类型参数中当年的月数，如 10，表示当年的第 10 个月 |
| int TimeSeconds( datetime time) | 返回日期类型参数中的秒数，如 26，表示第 26 秒 |
| int TimeYear( datetime time) | 返回日期类型参数中的年份，如 2009，表示 2009 年 |
| int Year( ) | 返回当前服务器的年份，如 2010，表示 2010 年 |

## 2.2.6 蜡烛序列函数

我们经常需要计算 n~n+i 个蜡烛的最高最低价，因此这组函数用途十分广。

| | |
|---|---|
| iBars(NULL,0) | 获取当前图表中蜡烛总数 |
| iBarShift(NULL,0,D'2010.09.01') | 获取当前图表自 2010-9-1 以来的蜡烛总数 |
| iHighest(NULL,0,MODE_HIGH,20,4) | 获取从第 4 个蜡烛开始的 20 个蜡烛范围内最高价的蜡烛序号 |
| iLowest(NULL,0,MODE_LOW,20,4) | 获取从第 4 个蜡烛开始的 20 个蜡烛范围内最低价的蜡烛序号 |

## 2.2.7 交易函数

关于交易函数详见 MT4 的帮助，具体用法在本书后续的范例中会频繁出现。在这里需要强调的是：

1、在自定义指标中不能调用 OrderSend(), OrderClose, OrderCloseBy, OrderDelete 和 OrderModify 交易函数。

2、OrderClose, OrderCloseBy, OrderDelete 和 OrderModify 函数在调用前必须用 OrderSelect()命令选择订单。

## 2.2.8 数学、三角函数

关于数学函数详见 MT4 的帮助。

值得强调的是绝对值函数是用频率最高，我们经常需要判断当前价是否达到了预期的止盈止损，就要是用这个函数。下面是个例句：

```
If (MathABS(Close[0]-OrderOpenPrice())>StopLoss*Point;//如果价位达到止损
```

使用这个语句的意义就在于我们不必去管当前订单是买入类型还是卖出类型。

## 2.2.9 数组函数

关于数组函数详见 MT4 的帮助。在此强调以下几个注意事项：

1、 数组的最大维数最大为 4 维。

例如定义一个数组为 myArray[10,10,10,10]，说明该数组有 4 维，每维有 10 个元素。

2、 维数元素序号从 0 开始计算。

例如 myArray[0]，就表示变量 myArray 第 0 个位置的数据。假如该数组定义为 10 个数字，那么第 10 个数字就应该表示为 myArray[9]。

3、MQL4 不是专业的计算机开发语言，在数组使用方面有不严谨之处。比如在编写指标的时候，你预先定义了一个一维数组 A[]，这个方括号里为空表示可以使用任意多个元素，实际在调用这个数组的时候，必须先定义元素数量，否则无法取值。

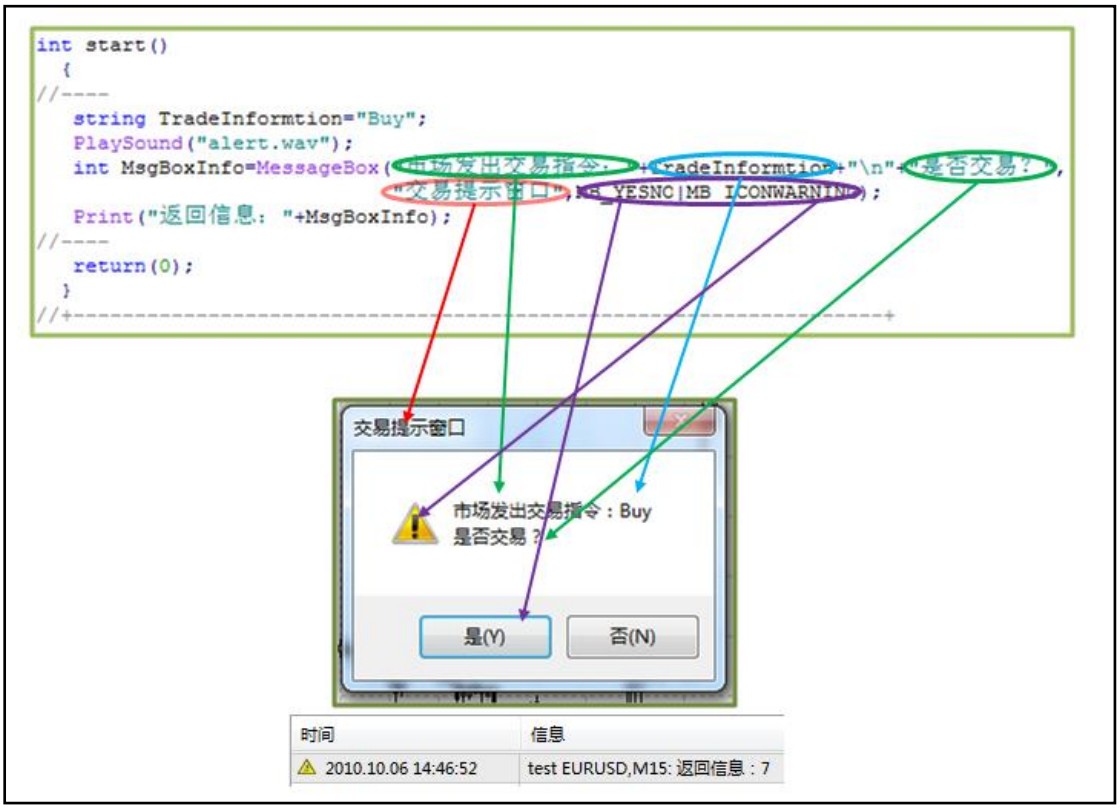
## 2.2.10弹出消息框函数

【源代码】

```
int start()
  {
//----
    string TradeInformtion="Buy";
    PlaySound("alert.wav");
    int MsgBoxInfo=MessageBox("市场发出交易指令："+TradeInformtion+"\n"+"是否交易？",
                              "交易提示窗口",MB_YESNO|MB_ICONWARNING);
    Print("返回信息："+MsgBoxInfo);
//----
```

```
    return(0);
  }
```

【源代码说明】



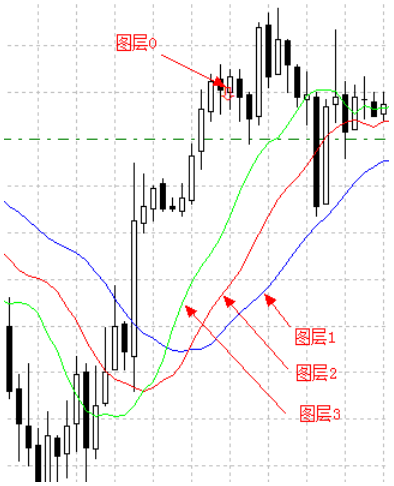

MessageBox 需要调用 mql 的函数，因此在程序头需要添加一个语句，否则通不过编译，该语句后面不要跟“;”。

```
#include <WinUser32.mqh>
```

## 2.3 自定义指标

技术指标是一种用来辅助判断行情的程序，按照特定的算法经过对市场数据计算后的值在屏幕上用线条、箭头等标注出来。

MQL4 规定在同一个图标中最多只能画 8 种类型的线条或者符号，为了方便理解，我们在此称为 8 个图层。如下图：




自定义指标又分为两种类型，一个是在主图中显示，如移动平均线，一个是在副图中显示，如 MACD。

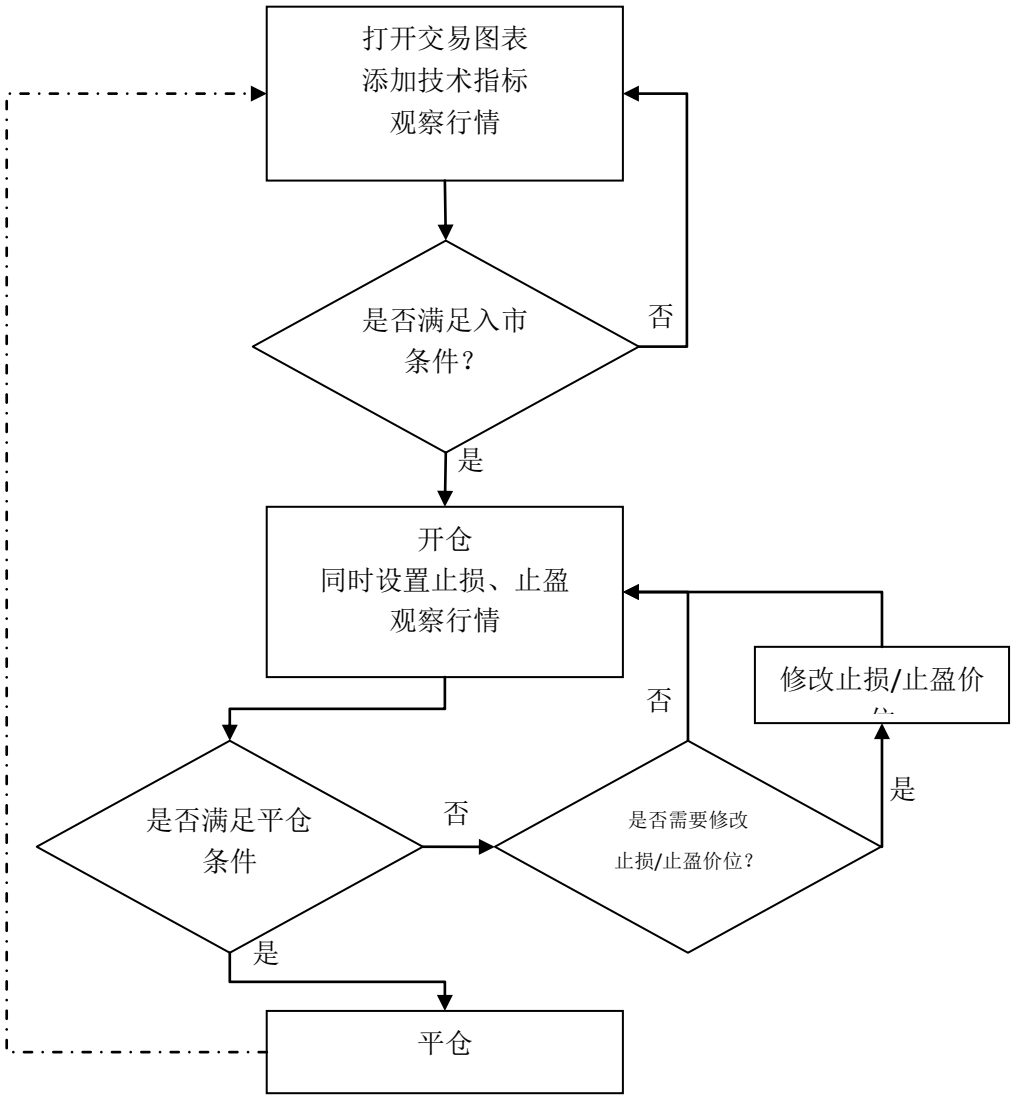
在本书的范例中有一个指标的源代码，通过理解源代码比任何论述都有效。

# 第三章　编程进阶

## 3.1 构思策略

### 3.1.1 交易过程的说明

下面是一个完整的交易流程图：




毫无疑问，所有的人都会按照上面这个流程进行外汇交易，同时得到一个结果：盈利或者亏损。在交易过程中我们会根据技术指标提供的信号决定买入或者卖出，再根据技术指标提供的信号决定修改止损止盈点，最后决定平仓出市。

相信所有的人都有一个共同的经历：当账面出现浮动盈利的时候，会认为盈利将继续扩大而没有按照计划获利平仓。当行情反向运行，盈利缩减的时候就会安慰自己，行情会掉头的再等等，又没有及时获利平仓。行情往往会跟你的美好愿望相悖而驰，当账面出现亏损抵达止损点的时候，依然梦想行情回头，甚至安慰自己说没关系，这一点点我亏得起，结果自然不言而喻。

每位参与外汇交易的人都有一套自己熟悉的指标体系来辅助决策，除此之外还有一套适合自己的资金盘子计划。每一次做单都需要考虑账户保证金和下单量，开仓后出现的浮动亏损与盈利情况又成为了下一步动作的重要参考，怎么重新设置止损止盈价位，用多大的补仓量等等，所有这些思考和行动的目的仅仅是确保账户资金的安全，实现稳步盈利，避免出现爆仓。咱们的老祖宗说过：留得青山在不愁没柴烧。

相信所有的人都知道要按照计划执行操作，但往往决策的时候忘记了计划，这就是人性

的弱点，谁都克服不了，包括我也逃脱不了，我经常这样评价自己和中国的汇友：亏得起，赢不起。因此，我们不难得出这个结论：使用 EA，能够回避人性的弱点，让操盘更加标准，更加严格按照计划执行。

从交易流程图的分析我们发现，一旦确定了技术指标、开仓量、补仓量、止损价位、止盈价位等等计划后，就是按照交易逻辑执行了，全过程完全可以不需要人工参与，证明 EA 可以帮助我们自动盯盘，根据制定好的策略执行开仓、平仓、挂单、修改止损止盈价位等等各种动作，是完全可行的。

我们在构思策略时最少要综合考虑以下三个方面：

- 价：入市的价位、止损止盈的价位
- 量：根据账户余额决定开仓、补仓的量
- 信号：根据技术指标决定入市（出市）及其方向

现在以“红狼教材-EURUSD-M30”为例，开始构思策略。

## 3.1.2 技术指标的选择

但凡炒外汇的人都会使用一些技术指标并将其整合，作为判断入市出市的参考依据。MQL4 语言提供了 29 个默认技术指标，囊括了几乎所有常见使用的指标。网上也有人提供 1000 个技术指标的，技术指标的作用是提供判断依据，我们几乎没有必要过多了解和学习默认值指标以外的，也不必深入钻研技术指标是怎么编制的，只要懂得技术指标是否发出了操作信号即可。

关于技术指标，可以参考我编写的《轻松研读 MT4 技术指标》一文。

## 3.1.3 风险控制的策略

对行情走势的判断之后，我们需要着重考虑风险控制。是重仓入市还是轻仓入市都是有讲究的，你不能输了一单就疯狂加倍反向做单，那样只会加快你账户爆仓。

## 3.1.3.1开仓下单量

开仓下单量计算公式如下：

$$开仓下单量=\frac{账户余额\times风险系数}{1标准手交易量}$$

说明：

杠杆　1：100；

1 标准手交易量为 125000 美元；

风险系数可根据自己的承受能力设置，通常我们设定风险系数为 5，系数大风险越高。

假设账户余额为 10000 美元，列表计算如下：

| 风险系数 | 下单量（手） |
|---|---|
| 1 | 0.08 |
| 2 | 0.16 |
| 3 | 0.24 |

| | |
|---|---|
| 4 | 0.32 |
| 5 | 0.40 |
| 6 | 0.48 |
| 7 | 0.56 |
| 8 | 0.64 |
| 9 | 0.72 |

### 3.1.3.2补仓下单量

在交易过程中，如果行情方向正确，账户可用保证金会随着增加，为了不浪费一轮好的行情，我们需要做补仓处理，以赚取更大的利润。或者行情出现了反向，为了减少亏损，加大盈利概率，也可以考虑反向补仓。

补仓量的大小是根据账户净额来确定的，如果账户净额大于账户余额，说明账面盈利，补仓量可以稍微加码，反之则需要减少。

计算补仓下单量也设置一个系数，计算公式如下：

$$\text{补仓下单量}=\text{开仓下单量}\times\left(1-\frac{\text{亏损订单数量}}{\text{补仓系数}}\right)$$

例如，补仓系数为 3，亏损订单数量为 1，那么这时补仓下单量就是开仓下单量的 2/3。

在后面的逻辑分析章节中，会禁止该公式出现负数，也会处理补仓系数为 0（分母为零）的情况，否则在程序运行时会出现错误。

### 3.1.3.3价格波动控制

根据技术指标我们发现了入市信号，根据帐户余额我们选定了下单量，就可以开仓了，此时止损止盈价格的设置是必须的，特别当你启动了 EA 后离开汇市，就显得更加重要。

考虑到汇市变化多端，风险难以控制，红狼教材-以 M30 为最小时间周期来考虑操作策略的，目的就是为了排除小周期（M1、M5、M15）市场出现的干扰信号。当然这只是经验数据，如果你的账户是 Mini 型的，杠杆又大于 100，那么就要因地制宜考虑参数的设置。

纵观外汇数据图表不难发现盘整行情多于单边行情，那么我们就需要利用趋势类指标确定单边行情的到来，同时利用震荡类指标过滤掉窄幅震荡行情。

控制价格波动没有绝对的区间，这是个见仁见智的数据。

## 3.2 逻辑分析

谈及逻辑执行，这可是计算机程序的强项，一个制定好的逻辑程序交给计算机要比人工的执行力强得多。

随着外汇 EA 化程度越来越高，许多人开始研究人工智能的计算模型，试图让计算机具备学习能力，来对付千变万化的汇市。最近类似网格、云计算等等人工智能专业术语充斥了整个 EA 世界。

我们不是专家，我们的目的是充分利用计算机的逻辑执行能力来辅助我们的决策，这就

简单了。

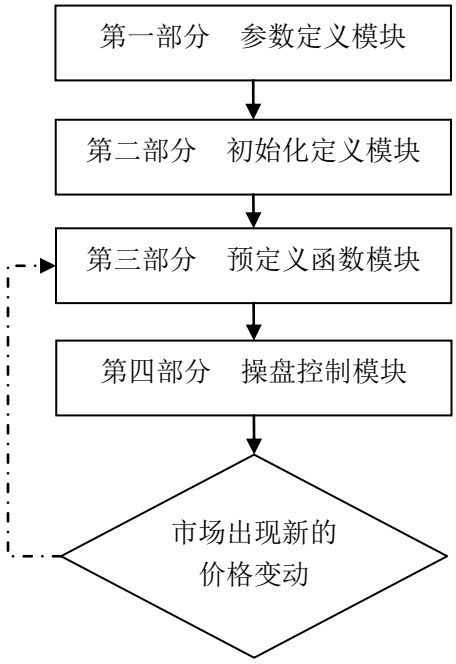
上一章针对外汇交易流程及风险控制的论述可以得到这么一个结论：外汇交易行为中有99%是逻辑行为，剩下的 1%是突发性事件因素，而对付突发性事件的解决方案就是设置合理的能够承受的止损空间，这仍然可以归类到逻辑行为。

本章着重针对交易行为和交易策略进行逻辑化的程序化的分析，旨在为下一章编制代码拟定一个准确详细的流程。

学会流程分析是编程的必要条件。

## 3.2.1 EA 逻辑框架

MQL4 语言为 EA 制定了一个固定的框架，见下图：



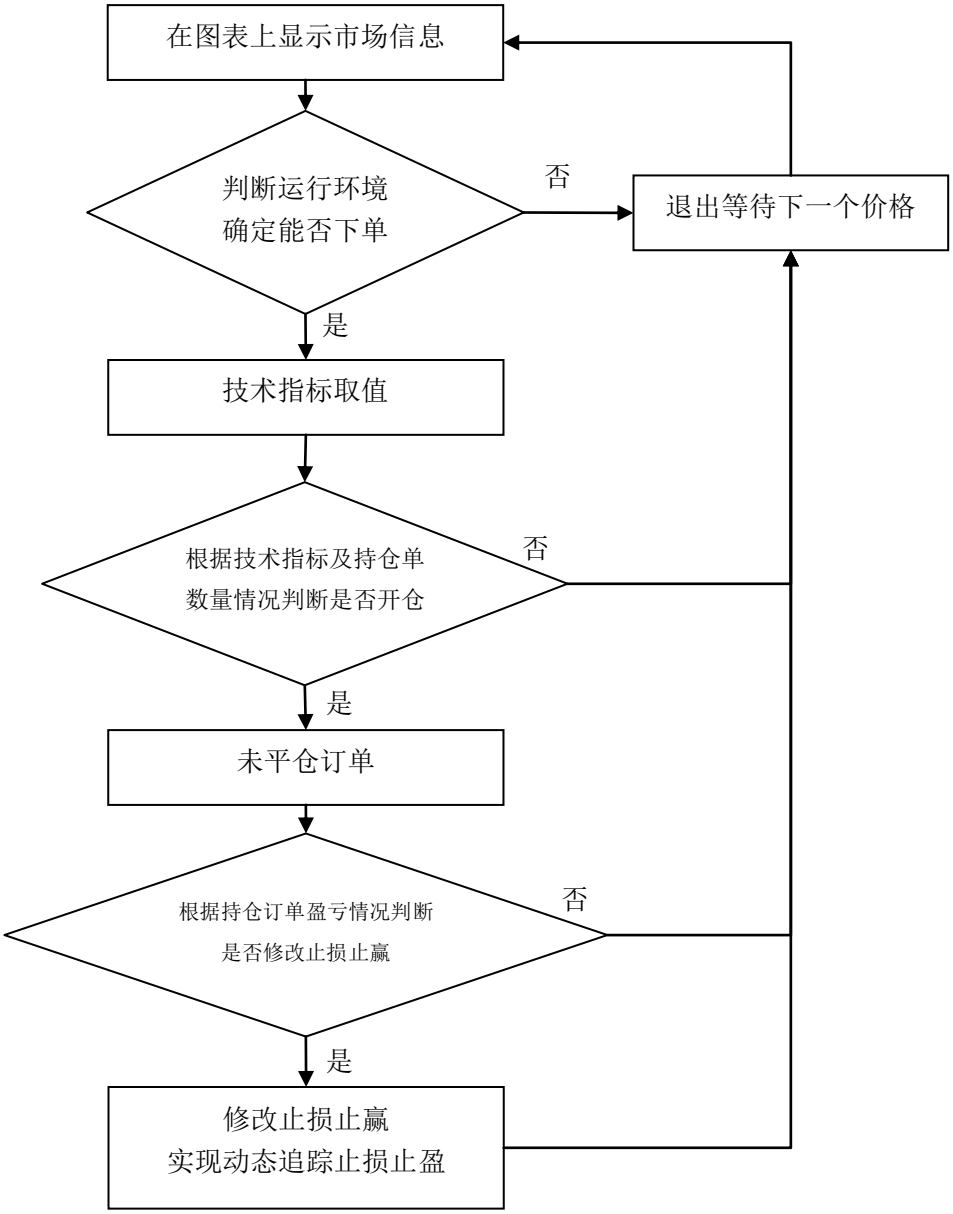

参数定义模块放置当前 EA 的属性，包括 EA 运行前需要人工定义的一些市场必须的参数（如止损、止盈点等），还可以包括一些外部函数库的调用和图表基本属性（如线型、颜色等）的定义。

初始化定义模块在 EA 运行时会先执行一次，一般用于进行和图表有关的一些属性的设置，也可以对后续程序中需要调用的变量给出初始值。

预定义函数模块在策略参数被修改后会执行一次，紧接着再执行初始化定义模块，策略首次导入图表时不执行该模块代码。

操盘控制模块是 EA 主模块，当市场出现每一次价格变动时都会执行一次。

## 3.2.2 操盘控制模块流程图




细心的人会发现，上面这个流程图中居然没有平仓的动作？这是个有趣的话题，另外找时间慢慢思考回味吧。

## 3.3 历史数据回测

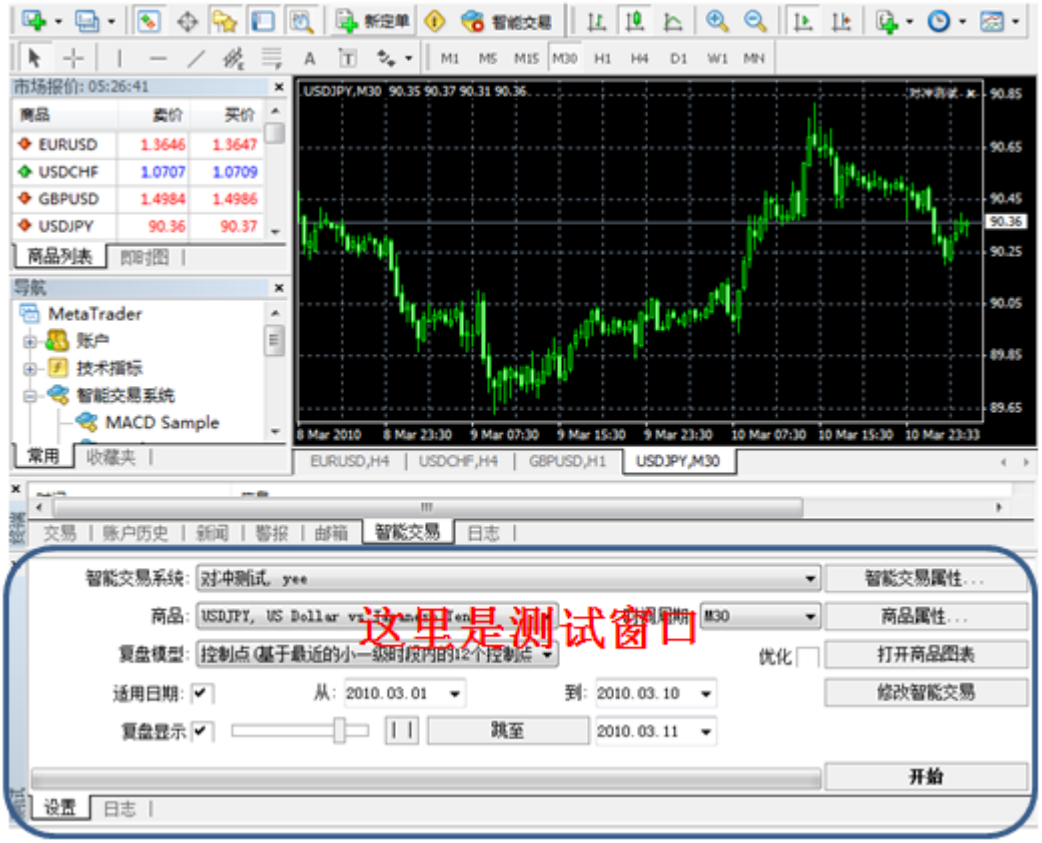
历史数据回测是自动化交易验证 EA 程序逻辑的一个很重要的环节。

MT4 提供了一个功能强大的系统测试模块，利用历史数据测试 EA 策略的结果并提交一份详细的测试报告，你可以根据报告调整 EA 的策略和参数，反复进行，以期达到最佳的模式。

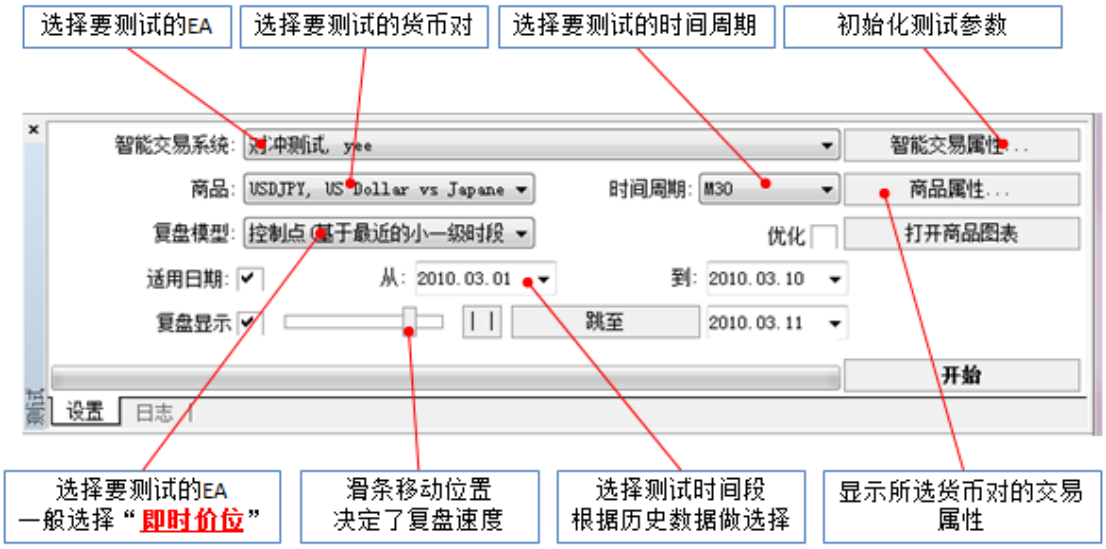
历史数据包含了开盘价、收盘价、最高价、最低价、成交量、时间等 6 项指标，分为 M1、M5、M15、M30、H1 、H4、D1、W1、MN 等 9 个周期。

## 3.3.1 开始一个 EA 测试

1、选择一个 EA 加载到图表
2、点击工具栏的“智能交易”停止智能交易
3、按 F6，打开测试窗口

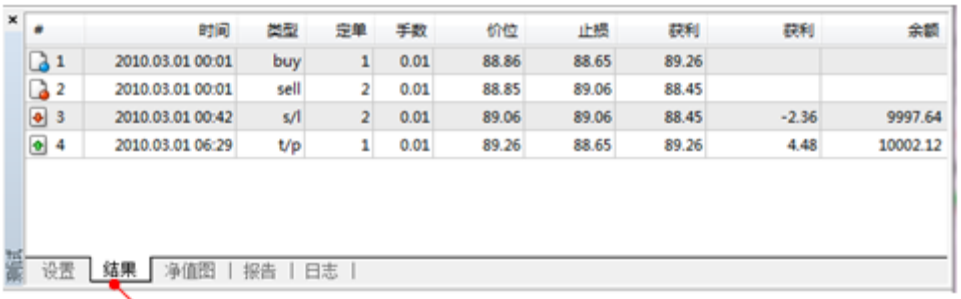


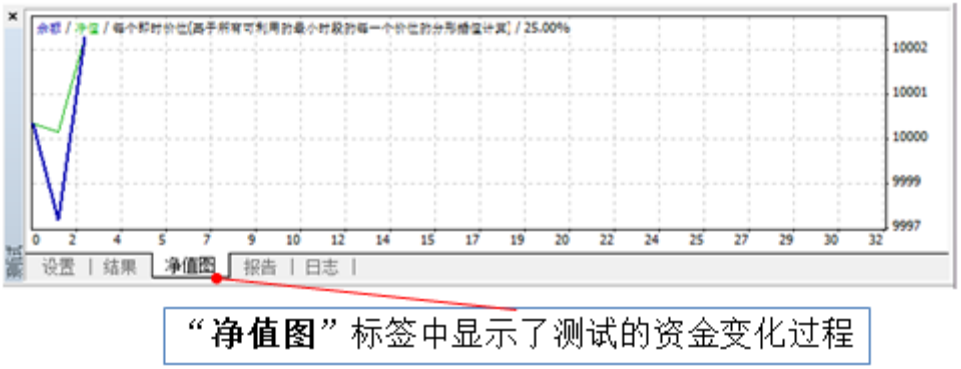

测试窗口各项说明如下：

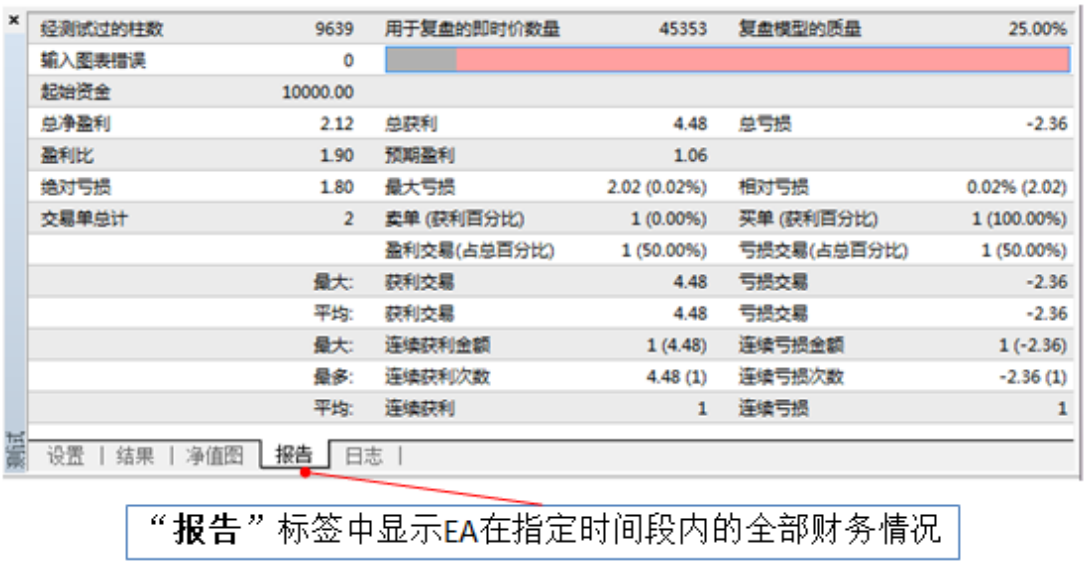



完成所有设置后，按“开始”，系统即开始测试：


测试完毕后，我们可以注意查看测试报告标签：





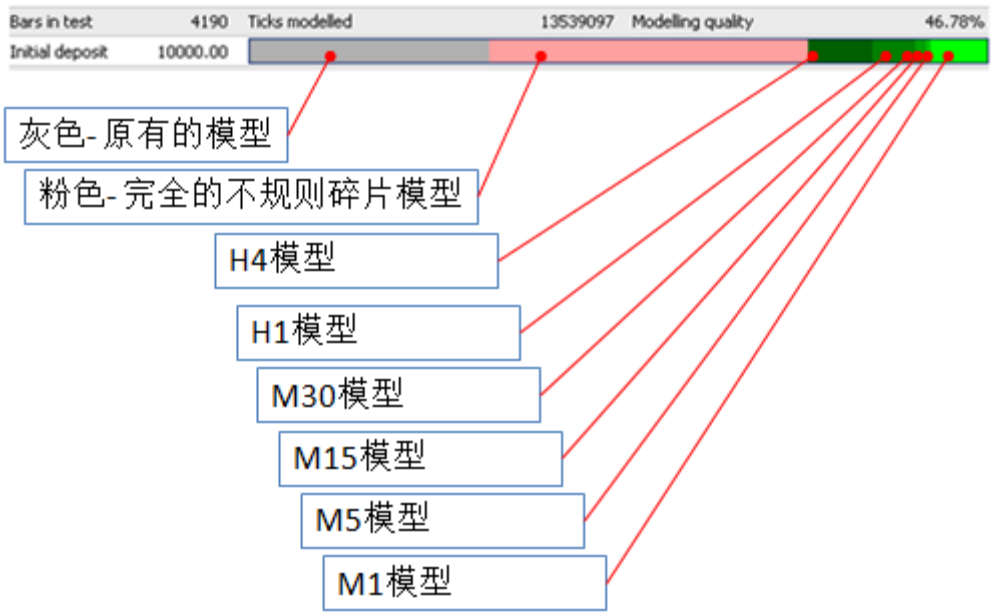

## 3.3.2 测试报告中各项指标说明

| | |
|---|---|
| 测试柱数<br>Bars in test | 历史数据蜡烛的总数 |
| 即时价数量<br>Ticks modelled | 历史数据最小模型是 M1，包含了 4 个即时价格（开盘价、收盘价、最高价、最低价），这 4 个价格用来模拟市场在 1 分钟内发出了 4 个新价格（tick）。因此，M5 时间周期每个蜡烛就包含了 20 个即时价位。该指标表示在制定时间周期内即时价位总数。 |
| 复盘模型的质量<br>Modelling quality | ModellingQuality = ((0.25*(StartGen-StartBar) + 0.5 *(StartGenM1-StartGen) + 0.9 *(HistoryTotal-StartGenM1)) / (HistoryTotal-StartBar))*100%;<br>其中：<br>HistoryTotal 限定时间段里历史数据蜡烛总数<br>StartBar 开始测试蜡烛的数，如果测试数据从图表的第一个蜡烛开始，则总数减去 101<br>StartGen 设定测试时间段内开始的蜡烛序数<br>StartGenM1 设定测试时间段内开始的 1 分钟蜡烛序数<br>对于最近时间范围数据库模型的开始和最近时间范围数据模型的开始存在重量系数 0. 25 的区别;<br>对于最近时间范围数据库模型的开始和最近时间范围数据模型的开始在原有分钟内存在重量系数 0. 5 的区别；<br>在原有时间上模型的开始和历史数据的末尾之间重量系数 0.9 的区别。 |
| 总净盈利<br>Total net profit | 净赢利值和净亏损值之间的差<br>TotalNetProfit = GrossProfit - GrossLoss |
| 总获利<br>Gross profit | 所有赢利交易总数的净赢利值 |

| | |
|---|---|
| 总亏损<br>Gross loss | 所有亏损交易总数的净亏损值 |
| 盈利比<br>Profit factor | 在设定测试时间内净赢利值与净亏损值的比<br>ProfitFactor = GrossProfit / GrossLoss |
| 预期盈利<br>Expected payoff | 预期盈利使用以下公式进行计算:<br>Expected Payoff = (ProfitTrades / TotalTrades) * (GrossProfit / ProfitTrades) -<br>(LossTrades / TotalTrades) * (GrossLoss / LossTrades)<br>其中：<br>TotalTrades 交易总数;<br>ProfitTrades 赢利交易总数;<br>LossTrades 亏损交易总数;<br>GrossProfit 净赢利交易总数;<br>GrossLoss 净亏损交易总数 |
| 绝对亏损<br>AbsoluteDrawDown | AbsoluteDrawDown = InitialDeposit – MinimalBalance |
| 最大亏损<br>MaximalDrawDown | 最大借款值和当前最小借款值的最大差距：<br>MaximalDrawDown = Max of (Maximal Peak - next Minimal Peak)<br>最大借款百分比的比率等于最大借款和它的各自价值的商：<br>MaxDrawDown % = MaxDrawDown / its MaxPeak * 100%<br>在报告中显示的其他结果可以应用简单的数学方法计算 |
| 交易单总计<br>Total trades | 在测试里的交易总数 |
| 卖单获利百分比<br>Short positions (won %) | 卖空仓位总数额和其中赢利百分比<br>（卖空仓位/卖空仓位总数*100%） |
| 买单获利百分比<br>Long positions (won %) | 看涨仓位总数额和其中赢利百分比<br>（看涨仓位/看涨仓位总数*100%） |
| 盈利交易（占总百分比）<br>Profit trades (% of total) | 赢利交易总数和交易总数的百分比<br>（赢利交易/交易总数*100%） |
| 亏损交易（占总百分比）<br>Loss trades (% of total) | 亏损交易总数和交易总数的百分比<br>（亏损交易/交易总数*100%） |
| 最大获利交易<br>Largest profit trade | 赢利交易中获得的最大获利 |
| 最大亏损交易<br>Largest loss trade | 亏损交易中获得的最大亏损 |
| 平均获利交易<br>Average profit trade | 赢利交易中赢利的平均数<br>(净赢利值 / 赢利交易) |
| 平均亏损交易<br>Average loss trade | 亏损交易中亏损的平均数<br>(净亏损值 / 亏损交易) |
| 最大连续获利金额<br>Maximum consecutive wins (profit in money) | 赢利总数和交易的赢利系列中最大连续盈利 |
| 最大连续亏损金额 | 亏损总数和交易的亏损系列中最大连续损失 |

| | |
|---|---|
| Maximum consecutive losses (loss in money) | |
| 最多连续获利次数 Maximal consecutive profit (count of wins) | 在交易总数中最大连续交易的赢利 |
| 最多连续亏损次数 Maximal consecutive loss (count of losses)） | 在交易总数中最大连续交易的赢利 |
| 平均连续获利数 Average consecutive wins | 赢利系列中连续盈利的平均数 |
| 平均连续亏损数 Average consecutive losses | 亏损系列中连续损失的平均数 |

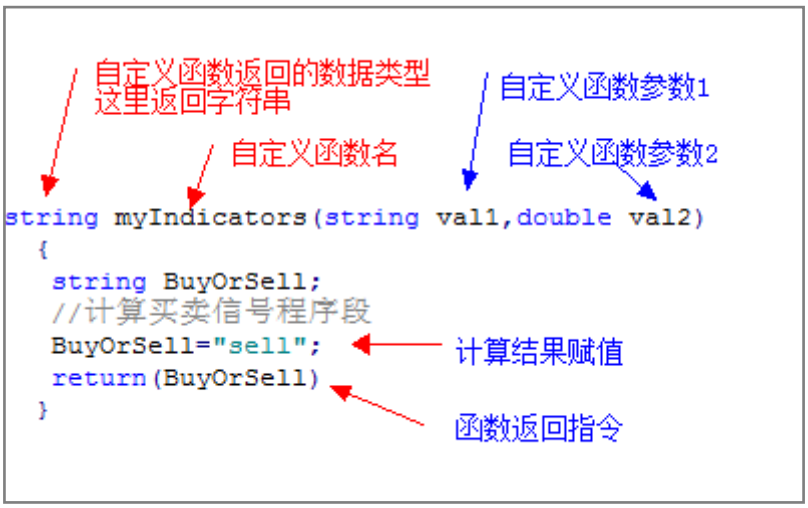
### 3.3.3 报告中色彩的含义




## 3.4 常用自定义函数

MQL4 提供了大量的基本函数和语句，然而我们在编程中发现很多的对行情的判断与操作都是重复的，比如判断指标快慢线是否交叉，这就需要用到自定义函数来简化主程序。

自定义函数通常放在程序的后面，函数格式与说明如下：




在程序中调用该自定义函数的例子如下：

```
If (myIndicators("金叉",Close[0]))="Sell"
```

将自定义函数参数写进去，就会得到按预定算法返回一个结果。

## 3.4.1 最大开仓量计算

保证金的合理使用是风险控制的重要手段，因此计算最大开仓量就显得非常重要。在许多风险控制论述中都会有这么一段类似的文字描述："开仓量为余额的 5%"。其实这种说法极其的不准确，甚至会导致因开仓量控制不严格而带来无谓的亏损。

不同货币对的 1 标准手自由保证金是不同的，而且如果你有持仓订单，由于价格变化导致账户净值也在变化，那么开仓量（手）也会发生变化。

以下代码计算了在当前货币对、当前价格的前提下，使用全部自由保证金（本币元）的最大开仓量（手）：

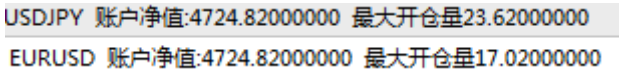
```
double myLots=(AccountEquity()/MarketInfo(Symbol(),MODE_MARGINREQUIRED));
```

其中 myBuyLots 为买入订单的最大开仓量（手），mySellLots 为卖出订单的最大开仓量（手）。计算方法是：

$$开仓量 = \frac{账户净值}{1标准手自由保证金}$$

所以，正确的 5%开仓量应该为：myLots×5%。

我们来看看通过程序计算显示的结果：




从上图可以看出，最大开仓量计算到了小数点后面 8 位，而实际操盘时的开仓量最小为 0.01 手，如果你直接使用这个数据，程序会报错，因此还需要通过内置函数将开仓量截止（不用四舍五入）到小数点后面 2 位：

```
myLots =NormalizeDouble(myLots,2)
OrderSend(Symbol(),OP_SELL, myLots ,Bid,0,0,0);//开一张卖出订单
```

## 3.4.2 新单开仓

读者也许很奇怪，系统中一条命令就能搞定，怎么还需要做这个函数呢？作者总结程序编写经验得出使用这个这个自定义函数能大大提高编程速度和质量。

在有些 ECN 平台上，利用 EA 新开仓是不允许设置止损止盈价的，在这里提醒读者，使用本函数尽量不要带止盈止损价格。

【函数代码】

```
/*
函数：新单开仓
      参数说明：
                开仓类型：Buy 买入订单、Sell 卖出订单、
                myLots 开仓量、myLossStop 止损点数、myTakeProfit 止盈点数
*/
void iOpenOrders(string myType,double myLots,int myLossStop,int myTakeProfit)
  {
   int mySPREAD=MarketInfo(Symbol(),MODE_SPREAD);//获取市场滑点
   double BuyLossStop=Ask-myLossStop*Point;
   double BuyTakeProfit=Ask+myTakeProfit*Point;
   double SellLossStop=Bid+myLossStop*Point;
   double SellTakeProfit=Bid-myTakeProfit*Point;
   if (myLossStop<=0)//如果止损参数为 0
      {
       BuyLossStop=0;
       SellLossStop=0;
      }
   if (myTakeProfit<=0)//如果止赢参数为 0
      {
       BuyTakeProfit=0;
       SellTakeProfit=0;
      }
   if (myType=="Buy")
       OrderSend(Symbol(),OP_BUY,myLots,Ask,mySPREAD,BuyLossStop,BuyTakeProfit);
   if (myType=="Sell")
       OrderSend(Symbol(),OP_SELL,myLots,Bid,mySPREAD,SellLossStop,SellTakeProfit);
  }
```

【调用语句说明】

```
iOpenOrders("Sell",0.1,25,40);
```

新单开仓只需要在函数后面跟 4 个参数，分别是交易类型（Buy 和 Sell）、开仓量、止损点数、止盈点数，四个参数的数据类型分别为 string、double、int、int。

例句中参数”Sell”表示开空头订单，0.1 表示开仓量为 0.1，25 为止损点数，40 为止盈点数。

如果止损、止盈点数都设置为 0，结果是新开订单不设置止损止盈。

## 3.4.3 持仓单平仓

在编程中，经常需要重复编写平仓代码，作者特意编写这个函数，只需要一条命令，就能实现多头订单、空头订单、盈利订单、亏损订单以及全部订单的平仓动作，大量减少了重复工作。

【函数代码】

```
/*
函数：持仓单平仓
      平仓类型：Buy 多头订单、Sell 空头订单、Profit 盈利订单、Loss 亏损订单、All 全部订单
*/
int CO_cnt;//订单计数器
void iCloseOrders(string myType)
  {
   if (OrderSelect(OrdersTotal()-1,SELECT_BY_POS)==false) return(0);//选择当前持仓订单
   if (myType=="All")
      {
       for(CO_cnt=OrdersTotal();CO_cnt>=0;CO_cnt--)
          {
           if(OrderSelect(CO_cnt,SELECT_BY_POS)==false) continue;
               else OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);
          }
      }
   if (myType=="Buy")//平掉所有多头订单
      {
       for(CO_cnt=OrdersTotal();CO_cnt>=0;CO_cnt--)
          {
           if(OrderSelect(CO_cnt,SELECT_BY_POS)==false) continue;
               else
                   if (OrderType()==0) OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);
          }
      }
   if (myType=="Sell")//平掉所有空头订单
      {
       for(CO_cnt=OrdersTotal();CO_cnt>=0;CO_cnt--)
          {
           if(OrderSelect(CO_cnt,SELECT_BY_POS)==false) continue;
               else
                   if (OrderType()==1) OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);
          }
      }
   if (myType=="Profit")//平掉所有盈利订单
      {
```

```
        for(CO_cnt=OrdersTotal();CO_cnt>=0;CO_cnt--)
           {
            if(OrderSelect(CO_cnt,SELECT_BY_POS)==false) continue;
                else
                    if (OrderProfit()>0) OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);
           }
       }
    if (myType=="Loss")
       {
        for(CO_cnt=OrdersTotal();CO_cnt>=0;CO_cnt--)
           {
            if(OrderSelect(CO_cnt,SELECT_BY_POS)==false) continue;
                else
                    if (OrderProfit()<0) OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);
           }
       }
  }
```

【调用语句说明】

```
iCloseOrders("All");
```

持仓单平仓只需要在函数后面跟一个参数，参数类型为 String。

参数规定如下：

Buy-多头订单、Sell-空头订单、Profit-盈利订单、Loss-亏损订单、All-全部订单。

## 3.4.4 追踪止损

【函数代码】

```
/*
函数：移动止损
      参数说明：myStopLoss 预设止损点数
      功能说明：遍历所有持仓订单，当持仓单获利达到止损点数时，修改止损价位
*/
void iMoveStopLoss(int myStopLoss)
  {
   int MSLCnt;//订单计数器
   if (OrderSelect(OrdersTotal()-1,SELECT_BY_POS)==false) return(0);//选择当前订单
   if (OrdersTotal()>0)
      {
       for(MSLCnt=OrdersTotal();MSLCnt>=0;MSLCnt--)
        {
         if (OrderSelect(MSLCnt,SELECT_BY_POS)==false) continue;
             else
            {
             if (OrderProfit()>0 && OrderType()==0 &&
```

```
((Close[0]-OrderStopLoss())>((2*myStopLoss)*Point)))
                    {

OrderModify(OrderTicket(),OrderOpenPrice(),Bid-Point*myStopLoss,OrderTakeProfit(),0);
                    }
                 if (OrderProfit()>0 && OrderType()==1 &&
((OrderStopLoss()-Close[0])>((2*myStopLoss)*Point)))
                    {

OrderModify(OrderTicket(),OrderOpenPrice(),Ask+Point*myStopLoss,OrderTakeProfit(),0);
                    }
                }
            }
        }
    }
```

【调用语句说明】

```
iMoveStopLoss(25);
```

止损点为 25，当订单盈利超过 25 点，函数自动修改持仓订单的止损价位。该函数将对所有的持仓订单进行操作。

【温馨提示】

对赌平台的常用手法之一就是：如果你在持仓订单上设置了止损价，他们就会通过服务器发出瞬间的数据导致你亏损平仓，然后再恢复正常的价格传送。因此，作者建议尽量不要对持仓订单设置止损价格，而是用程序计算平仓。

## 3.4.5 定时交易

【函数代码】

```
/*
函数：交易时间控制
        参数说明：开始自动交易时、分、停止交易时、分
        返回值：true 为自动交易有效，false 为自动交易无效
        备注：时分参数均为系统时间，不是北京时间
*/
bool EA.Valid=false;//该变量需要在程序头定义
bool iTimeControl(int myStartHour,int myStartMinute,
                          int myStopHour,int myStopMinute)
  {
    if (Hour()==0 && Minute()==0) EA.Valid=false;//新的一天变量初始化
    if (Hour()==myStopHour && Minute()==myStopMinute+1)//满足结束时间条件
       {
         EA.Valid=false;
       }
    if (Hour()==myStartHour && Minute()==myStartMinute)//满足开始时间条件
```

```
        {
          EA.Valid=true;
        }
     return(EA.Valid);
   }
```

【调用语句说明】

iTimeControl(15,00,17,30);//从系统时间 15:00~17:30 开始自动交易，函数返回 true

在程序中就可以这么使用：

```
If (iTimeControl(15,00,17,30))
 {
  //规定时间段内自动交易的程序代码
 }
```

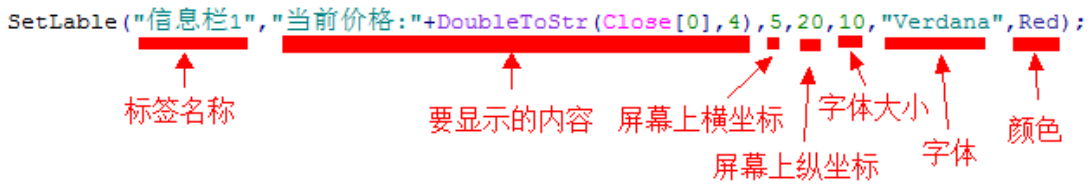
## 3.4.6 在屏幕上显示文字

【函数代码】

```
/*
函数：在屏幕上显示标签
参数说明：LableName：标签名称；LableDoc：文本内容；LableX：标签 X 位置；LableY：标签 Y 位置；DocSize：文本字号；DocStyle：文本字体；DocColor：文本颜色
*/
void iSetLable(string LableName,string LableDoc,int LableX,int LableY,
                    int DocSize,string DocStyle,color DocColor)
    {
    ObjectCreate(LableName, OBJ_LABEL, 0, 0, 0);
    ObjectSetText(LableName,LableDoc,DocSize,DocStyle,DocColor);
    ObjectSet(LableName, OBJPROP_XDISTANCE, LableX);
    ObjectSet(LableName, OBJPROP_YDISTANCE, LableY);
    }
```

【调用语句说明】

iSetLable("信息栏 1","当前价格:"+DoubleToStr(Close[0],4),5,20,10,"Verdana",Olive);




注意在要显示的内容中使用了 DoubleToStr()命令，控制显示小数点后 4 位，否则会显示小数点后面 8 位数字。

【显示效果】


这个函数与 MQL4 内置的 print 和 comment 不同，能自定义显示的位置、大小、颜色，对于美化界面，实时追踪数据很有帮助。

## 3.4.7 两点之间画线

【函数代码】

```
/*
函数：两点之间画线
参数说明： myFirstTime 第一点时间,myFirstPrice 第一点价格,mySecondTime 第二点时间,mySecondPrice 第二点价格
*/
int LineNo=0;
void    iDrawLine (int myFirstTime,double myFirstPrice,int mySecondTime,double mySecondPrice)

    {
      string myObjectName="Line"+LineNo;

ObjectCreate(myObjectName,OBJ_TREND,0,myFirstTime,myFirstPrice,mySecondTime,mySecondPrice);
       ObjectSet(myObjectName,OBJPROP_COLOR,Green);
       ObjectSet(myObjectName,OBJPROP_STYLE,STYLE_DOT);
       ObjectSet(myObjectName,OBJPROP_WIDTH, 1);
       ObjectSet(myObjectName,OBJPROP_BACK,false);
       ObjectSet(myObjectName,OBJPROP_RAY,false);
       i++;
    }
```

【调用语句说明】

iDrawLine (13,close[13],6,close[6]);//从第 13 个蜡烛的收盘价到第 6 个蜡烛的收盘间画一个绿色虚线

## 3.4.8 标注符号

【函数代码】

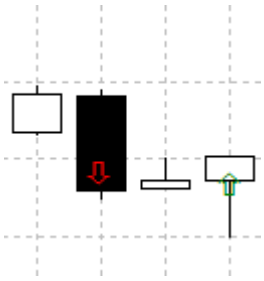
```
/*
函数：标注符号
        红箭头为卖出，绿箭头为买入，红绿圆圈为其他标记
        参数说明：mySignal 变量包括 Buy-买入箭头
                                    Sell-卖出箭头
                                    GreenMark-绿色圆圈
                                    RedMark-红色圆圈
                    myPrice 当前价格，符号标注位
*/
void iDrawSign(string mySignal,double myPrice)
  {
   if (mySignal=="Buy")
       {
        ObjectCreate("BuyPoint-"+Time[0],OBJ_ARROW,0,Time[0],myPrice);
        ObjectSet("BuyPoint-"+Time[0],OBJPROP_COLOR,Green);
        ObjectSet("BuyPoint-"+Time[0],OBJPROP_ARROWCODE,241);
       }
   if (mySignal=="Sell")
       {
        ObjectCreate("SellPoint-"+Time[0],OBJ_ARROW,0,Time[0],myPrice);
        ObjectSet("SellPoint-"+Time[0],OBJPROP_COLOR,Red);
        ObjectSet("SellPoint-"+Time[0],OBJPROP_ARROWCODE,242);
       }
   if (mySignal=="GreenMark")
       {
        ObjectCreate("GreenMark-"+Time[0],OBJ_ARROW,0,Time[0],myPrice);
        ObjectSet("GreenMark-"+Time[0],OBJPROP_COLOR,Green);
        ObjectSet("GreenMark-"+Time[0],OBJPROP_ARROWCODE,162);
       }
   if (mySignal=="RedMark")
       {
        ObjectCreate("RedMark-"+Time[0],OBJ_ARROW,0,Time[0],myPrice);
        ObjectSet("RedMark-"+Time[0],OBJPROP_COLOR,Red);
        ObjectSet("RedMark-"+Time[0],OBJPROP_ARROWCODE,162);
       }
  }
```

【调用语句说明】

iDrawSign(“Buy”,close[6]);//在第 5 个蜡烛收盘价上画一个买入箭头

在当前价格标注“买入”箭头。

【显示效果】



## 3.4.9 指标线交叉信号

【函数代码】

```
/*
函数：快慢指标线交叉信号
        参数说明：myFast0 当前蜡烛快线值
                  mySlow0 当前蜡烛慢线值
                  myFast1 前个蜡烛快线值
                  mySlow1 前个蜡烛慢线值
        返回值：UpCross 上穿、DownCross 下穿、N/A 无穿越
*/
string iCrossSignal(double myFast0,double mySlow0,double myFast1,double mySlow1)
   {
    string myCrossSignal="N/A";
    if (myFast0>mySlow0 && myFast1<=mySlow1) myCrossSignal="UpCross";//上穿越
    if (myFast0<mySlow0 && myFast1>=mySlow1) myCrossSignal="DownCross";//下穿越
    return(myCrossSignal);
   }
```

【调用语句说明】

```
double myMA5_0=iMA(Symbol(),0,5,0,MODE_SMA,PRICE_CLOSE,0);
double myMA5_1=iMA(Symbol(),0,5,0,MODE_SMA,PRICE_CLOSE,1);
double myMA10_0=iMA(Symbol(),0,10,0,MODE_SMA,PRICE_CLOSE,0);
double myMA10_1=iMA(Symbol(),0,10,0,MODE_SMA,PRICE_CLOSE,1);
string mySignal=iCrossSignal(myMA5_0,myMA10_0,myMA5_1,myMA10_1);
```

上面这段代码能够获取移动平均线快线（5 个蜡烛周期）与慢线（10 个蜡烛周期）交叉信号，变量 mySignal 返回交叉信号。

# 3.5 EA 范例 1 鳄鱼三线+Force

```
//+------------------------------------------------------------------+
//|                                                   红狼.mq4 |
//|                                                      laoyee |
//|                                                  QQ:921795 |
//+------------------------------------------------------------------+
/*
货币对：EUR/USD
时间周期：M30
技术指标：鳄鱼三线  8，5，3
```

```
            趋势指标+震荡指标
开仓条件：买入条件   鳄鱼三线成顺序
            用 Force 指标过滤盘整行情
平仓条件：鳄鱼线交叉
*/
#property copyright "laoyee"
#property link          "QQ:921795"
extern int StopLoss=40;
extern int TakeProfit=0;
extern double MaxRisk=30;//资金风险 1=1%
extern double Filter=0.35;//Force 指标过滤参数
double Alligator_jaw,Alligator_teeth,Alligator_lips,
          Envelops21_upper,Envelops21_lower,Force3;
int start()
  {
   OrderSelect(0,SELECT_BY_POS);//选当前订单
   //显示市场信息
   SetLable("时间栏","星期"+DayOfWeek()+"  市场时间："+Year()+"-"+Month()+"-"+Day()+" "+
               Hour()+":"+Minute()+":"+Seconds(),200,0,9,"Verdana",Red);
   SetLable("信息栏","市场信号："+ReturnMarketInfomation()+
               "       当前订单盈亏："+DoubleToStr(OrderProfit(),2),5,20,10,"Verdana",Blue);
   //周五 20 点停止交易，盈利订单平仓
   if (DayOfWeek()==5 && Hour()>=20 && Minute()>=0)
       {
       if (OrderProfit()>0) OrderClose(OrderTicket(),OrderLots(),Ask,0);
       return(0);
       }
   //新开仓订单时间不足一个时间周期，不做任何操作返回
   if (TimeCurrent() - OrderOpenTime() <=PERIOD_M30*60) return (0);
   double sl_buy=Ask-StopLoss*Point;
   double tp_buy=Ask+TakeProfit*Point;
   double sl_sell=Bid+StopLoss*Point;
   double tp_sell=Bid-TakeProfit*Point;
   if (StopLoss==0) {sl_buy=0;sl_sell=0;}
   if (TakeProfit==0) {tp_buy=0;tp_sell=0;}

   //开仓操作
   if (Symbol()=="EURUSD" && OrdersTotal()==0)//EURUSD 货币对，没有订单，则开仓
       {
       if (                                        )
           OrderSend(Symbol(),OP_BUY,LotsOptimized(MaxRisk),Ask,0,sl_buy,tp_buy);
       if (                                        )
           OrderSend(Symbol(),OP_SELL,LotsOptimized(MaxRisk),Bid,0,sl_sell,tp_sell);
       }

   //平仓操作
   if (OrderProfit()>0)//止盈操作
       {
       if (Symbol()=="EURUSD" && OrdersTotal()==1 && OrderType()==OP_BUY && ReturnMarketInfomation()=="DownCross")
           OrderClose(OrderTicket(),OrderLots(),Ask,0);
       if (Symbol()=="EURUSD" && OrdersTotal()==1 && OrderType()==OP_SELL && ReturnMarketInfomation()=="UpCross")
           OrderClose(OrderTicket(),OrderLots(),Bid,0);
       }
   if (OrderProfit()<0)//止损操作
       {
       if (Symbol()=="EURUSD" && OrdersTotal()==1 && OrderType()==OP_BUY && Alligator_lips<Alligator_jaw)
           OrderClose(OrderTicket(),OrderLots(),Ask,0);
       if (Symbol()=="EURUSD" && OrdersTotal()==1 && OrderType()==OP_SELL && Alligator_lips>Alligator_jaw)
           OrderClose(OrderTicket(),OrderLots(),Bid,0);
       }
   return(0);
  }

/*
函数：优化保证金，确定开仓量，进行风险控制
       根据风险值 RiskValue 计算开仓量
       如果出现亏损订单，则下一单开仓量减半
*/
double LotsOptimized(double RiskValue)
```

```
    {
    double  iLots=NormalizeDouble((AccountBalance()*RiskValue/100)/MarketInfo(Symbol(),MODE_MARGINREQUIRED),2);//最大可开仓手数
    if (iLots<0.01) {iLots=0;Print ("保证金余额不足！ ");}
    OrderSelect(OrdersHistoryTotal()-1,SELECT_BY_POS,MODE_HISTORY);
    if (OrderProfit()<0) iLots=0.01;//上一个订单亏损，本次开仓量减半
    return (iLots);
    }
/*
函数：在屏幕上显示标签
        LableName：标签名称；LableDoc：文本内容；LableX：标签 X 位置；LableY：标签 Y 位置；
        DocSize：文本字号；DocStyle：文本字体；DocColor：文本颜色
*/
void SetLable(string LableName,string LableDoc,int LableX,int LableY,
                    int DocSize,string DocStyle,color DocColor)
    {
    ObjectCreate(LableName, OBJ_LABEL, 0, 0, 0);
    ObjectSetText(LableName,LableDoc,DocSize,DocStyle,DocColor);
    ObjectSet(LableName, OBJPROP_XDISTANCE, LableX);
    ObjectSet(LableName, OBJPROP_YDISTANCE, LableY);
    }
/*
函数：返回市场信息
        获取技术指标参数，通过比对，返回市场信息：
        Buy-买入信号，sell-卖出信号，Rise-涨势行情，Fall-跌势行情，
        UpCross-向上翻转，DownCross-向下反转,反转信号为平仓信号
*/
string ReturnMarketInfomation()
    {
    string MktInfo="N/A";
    //读取指标数值
    Alligator_jaw=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORJAW,0),4);

Alligator_teeth=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORTEETH,0),4);
    Alligator_lips=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORLIPS,0),4);
    double
Alligator_jaw_1=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORJAW,1),4);
    double
Alligator_teeth_1=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORTEETH,1),4);
    double
Alligator_lips_1=NormalizeDouble(iAlligator("EURUSD",30,8,0,5,0,3,0,MODE_EMA,PRICE_WEIGHTED,MODE_GATORLIPS,1),4);
    Force3=NormalizeDouble(iForce("EURUSD",30,3,MODE_EMA,PRICE_WEIGHTED,0),4);
    //指标分析，返回市场信息 || Force3<-FilterForce3>Filter ||
    if (Alligator_lips>Alligator_teeth && Alligator_lips_1<=Alligator_teeth_1)
        MktInfo="UpCross";
    if (Alligator_lips<Alligator_teeth && Alligator_lips_1>=Alligator_teeth_1)
        MktInfo="DownCross";
    if (Alligator_lips>Alligator_teeth && Alligator_teeth>Alligator_jaw)
        MktInfo="Rise";
    if (Alligator_lips<Alligator_teeth && Alligator_teeth<Alligator_jaw)
        MktInfo="Fall";
    if (Force3>Filter && MktInfo=="Rise")
        MktInfo="Buy";
    if (Force3<-Filter && MktInfo=="Fall")
        MktInfo="Sell";
    return(MktInfo);
    }
```

## 3.6 EA 范例 2 MACD 与补仓

```
//+------------------------------------------------------------------+
//|                                                Laoyee-MACD.mq4 |
//|                                                         laoyee |
//|                                                  QQ:921795 |
//+------------------------------------------------------------------+
/*
货币对：EURUSD
时间周期：M15
技术指标：MACD 10,50
开仓条件：MACD 当前柱大（小）于前第 n 柱，MACD 当前值大（小）于 0，并且价格也高于（低于）前第 n 柱，开多（空）仓
加仓条件：满足开仓条件，且建仓价大（小）于前一单建仓价，加多（空）仓
          再次加仓，满足加仓条件做以下操作：
          1、如果新订单盈利，加仓单开仓量恢复正常
          2、如果新订单亏损，加仓单开仓量再加倍
          3、设定开仓量 0.01 手，翻倍补仓条件为余额盈利 n%，每盈利 n%，开仓手数翻倍

平仓条件：当前柱小（大）于前第 n 柱，并且价格也低于（高于）前第 n 柱，平所有订单
止损止盈：多空互换，不间断交易
资金控制：如果出现亏损，下一单开仓量加倍，加倍仓盈利，下一单按正常量建仓
          按账户余额一定的比例（n%）计算保证金比例，确定开仓量。
*/
#property copyright "laoyee"
#property link      "QQ:921795"
extern double Init_Lots = 0.01;
extern double Profit_Rate = 34;
extern double Max_Bet_Lots = 0.4;
extern double LostRate = 2;

int Order_Total = 50;
int Bet_Order = 50;
bool LotsDouble = true;
int iBet_Order = 0;//加倍订单计数器变量
double perProfit = 0;//每批订单总盈亏变量
double iLots;//追加订单开仓量变量
double Init_Balance;//账户初始余额变量
double xMax_Bet_Lots;//最大浮动开仓量变量
int TicketNo;
int init()
    {
```

```
      iLots = Init_Lots;
      xMax_Bet_Lots = Max_Bet_Lots;
      Init_Balance = AccountBalance();//取初始账户余额
    }

int start()
  {
    //提取市场信号
    string mktSignal=ReturnMarketInfomation();
    //显示市场信息
    SetLable("时间栏","星期"+DayOfWeek()+" 市场时间："+Year()+"-"+Month()+"-"+Day()+" "+
                Hour()+":"+Minute()+":"+Seconds(),200,0,9,"Verdana",Red);
    SetLable("信息栏","市场信号："+mktSignal+"    最大开仓量："
"+DoubleToStr(xMax_Bet_Lots,2),5,60,10,"Verdana",Blue);
    SetLable("信息栏 1","初始余额："+DoubleToStr(Init_Balance,2)+"    当前余额："
"+DoubleToStr(AccountBalance(),2),5,20,10,"Verdana",Blue);
    SetLable("信息栏 2","最低开仓量："+DoubleToStr(NewLots(),2)+"    浮动开仓量："
"+DoubleToStr(iLots,2),5,40,10,"Verdana",Blue);

    //新开仓
    if (OrdersTotal()==0)
        {
         if (perProfit<0 && LotsDouble==true) iLots=iLots*LostRate;//如果前一批订单盈利为负
数且允许亏损加倍，开仓量翻倍
          if (iLots>xMax_Bet_Lots) iLots=xMax_Bet_Lots;//限制最大开仓量
          Open_New_Order(iLots);
          perProfit=0;//前一批订单盈利变量清 0
        }
    //处理已有订单
    if (OrdersTotal()>0)
        {
         OrderSelect(OrdersTotal()-1,SELECT_BY_POS);//选择当前订单
         //新开仓订单时间不足一个时间周期，不做任何操作返回
         if (TimeCurrent() - OrderOpenTime() <=Period()*60) return (0);

         //追加盈利订单
         double iiLots=iLots;
         if (iBet_Order>Bet_Order-1) iiLots=NewLots();//如果超过加倍订单数量，交易量恢复初
始值
         if (OrderType()==0 && mktSignal=="Buy" && OrdersTotal()<=Order_Total &&
Ask>OrderOpenPrice())//追加买入订单
             {
              TicketNo=OrderSend(Symbol(),OP_BUY,iiLots,Ask,0,0,0);
              Draw_Mark(TicketNo);
```

```
                iBet_Order=iBet_Order+1;//加倍订单计数
               }
          if (OrderType()==1 && mktSignal=="Sell" && OrdersTotal()<=Order_Total &&
Bid<OrderOpenPrice())//追加卖出订单
               {
                TicketNo=OrderSend(Symbol(),OP_SELL,iiLots,Bid,0,0,0);
                Draw_Mark(TicketNo);
                iBet_Order=iBet_Order+1;//加倍订单计数
               }

          //止损平仓。如果出现反向信号，平掉所有订单
          if (OrderType()==0 && mktSignal=="Sell")
              {
               for(int G_Count=OrdersTotal();G_Count>=0;G_Count--)
                  {
                   if(OrderSelect(G_Count,SELECT_BY_POS)==false) continue;
                      else OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);//平仓
                   perProfit=perProfit+OrderProfit();//利润累加
                  }
               if (perProfit>=0) iLots = NewLots();//计算盈利后新开仓量
               iBet_Order=0;//加倍订单变量清零
              }
          if (OrderType()==1 && mktSignal=="Buy")
              {
               for(G_Count=OrdersTotal();G_Count>=0;G_Count--)
                  {
                   if(OrderSelect(G_Count,SELECT_BY_POS)==false) continue;
                      else OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),0);//平仓
                   perProfit=perProfit+OrderProfit();//利润累加
                  }
               if (perProfit>=0) iLots = NewLots();//计算盈利后新开仓量
               iBet_Order=0;//加倍订单变量清零
              }
         }
     return(0);
   }

/*
计算账户余额，返回最新开仓量(手数)
*/
double NewLots()
     {
      //计算翻倍倍数
      double xRate=((AccountBalance()-Init_Balance)/Init_Balance)/(Profit_Rate/100);//取倍率的
```

```
整数部分
      //计算开仓手数
      double xLots=NormalizeDouble(Init_Lots*xRate,2);
      if (xLots<0.01) xLots=0.01;//如果开仓量小于最小允许标准手数，按最小允许标准手数取值
      if (xRate>1) xMax_Bet_Lots = Max_Bet_Lots*xRate;//如果翻倍倍数大于 1，才计算最大浮动开仓量变量
      return(xLots);
    }

/*
根据市场信号开新仓，带开仓量参数
*/
void Open_New_Order(double MyLots)
    {
      if (ReturnMarketInfomation()=="Buy")
          TicketNo=OrderSend(Symbol(),OP_BUY,MyLots,Ask,0,0,0);
      if (ReturnMarketInfomation()=="Sell")
          TicketNo=OrderSend(Symbol(),OP_SELL,MyLots,Bid,0,0,0);
      Draw_Mark(TicketNo);
    }



/*
判断 MACD 以及市场价格，提交"Buy"和"Sell"信号
*/
string ReturnMarketInfomation()
    {
      string MktInfo="N/A";
      double MACD_0=iMACD(NULL,0,10,60,1,PRICE_CLOSE,MODE_SIGNAL,0);
      double MACD_2=iMACD(NULL,0,10,60,1,PRICE_CLOSE,MODE_SIGNAL,10);
      double price_0=Close[0];
      double price_high_2=High[2];
      double price_low_2=Low[2];
      if (MACD_0>(MACD_2+0.00003) && price_0>price_high_2 )
         {
          MktInfo="Sell";
         }
      if (MACD_0<(MACD_2-0.00003) && price_0<price_low_2 )
         {
          MktInfo="Buy";
         }
      return(MktInfo);
```

```
    }

/*
函数：在屏幕上显示标签
        LableName：标签名称；LableDoc：文本内容；LableX：标签 X 位置；LableY：标签 Y
位置；
        DocSize：文本字号；DocStyle：文本字体；DocColor：文本颜色
*/
void SetLable(string LableName,string LableDoc,int LableX,int LableY,
                    int DocSize,string DocStyle,color DocColor)
    {
    ObjectCreate(LableName, OBJ_LABEL, 0, 0, 0);
    ObjectSetText(LableName,LableDoc,DocSize,DocStyle,DocColor);
    ObjectSet(LableName, OBJPROP_XDISTANCE, LableX);
    ObjectSet(LableName, OBJPROP_YDISTANCE, LableY);
    }

/*
函数：在屏幕上做开仓标记，红色箭头为卖出订单，绿色箭头为买入订单
        MyTicket：订单号
*/

void Draw_Mark(int MyTicket)
    {
     string ArrowMyTicket="Arrow:"+DoubleToStr(MyTicket,0);
     OrderSelect(OrdersTotal()-1,SELECT_BY_POS);//选定当前订单
     //建立箭头模型
     int ArrowValue;color ArrowColor;
     if (OrderType()==0) {ArrowValue=221;ArrowColor=Green;}
     if (OrderType()==1) {ArrowValue=222;ArrowColor=Red;}
     ObjectCreate(ArrowMyTicket,OBJ_ARROW,0,Time[0],OrderOpenPrice());
     ObjectSet(ArrowMyTicket,OBJPROP_ARROWCODE,ArrowValue);//设置箭头，向上的箭头
221，向下的箭头 222
     ObjectSet(ArrowMyTicket,OBJPROP_COLOR,ArrowColor);//设置箭头颜色，买入为 Green，
卖出为 Red
    }
```

## 3.7 自定义指标范例：图形化回顾历史交易

这是一个笔者原创的一个很好用的指标，它可以将你过去的交易记录在主图中用不同的颜色标注箭头并连线，鼠标移动到箭头上能看到开仓平仓价，鼠标移动到线条上能看到交易单号，同时在主图的左上部分显示交易统计数据。如下图所示：


源代码如下，通过这个例子，你有机会很好理解指标的编写，同时熟练运用一些 MQL4 内置的函数。

```
//+------------------------------------------------------------------+
//|                                                 指标-交易历史图形化.mq4 |
//|                                                                  laoyee |
//|                                                                qq:921795 |
//+------------------------------------------------------------------+
#property copyright "laoyee"
#property link      "qq:921795"
//指标在主图显示
#property indicator_chart_window
#property indicator_buffers 3
//定义买入订单开盘箭头、卖出订单开盘箭头、平仓符号
double OpenBuyArrow[],OpenSellArrow[],CloseArrow[];

int init()
  {
   //设置开仓买入订单箭头模型
   SetIndexStyle(0, DRAW_ARROW,EMPTY,1,Green);
   SetIndexArrow(0, 221);
   SetIndexBuffer(0, OpenBuyArrow);
   SetIndexLabel(0,"买入订单开盘价");
   //设置开仓卖出订单箭头模型
   SetIndexStyle(1, DRAW_ARROW,EMPTY,1,Red);
   SetIndexArrow(1, 222);
   SetIndexBuffer(1, OpenSellArrow);
   SetIndexLabel(1,"卖出订单开盘价");
   //设置平仓订单符号模型
   SetIndexStyle(2, DRAW_ARROW,EMPTY,1,DarkOrange);
   SetIndexArrow(2, 251);
   SetIndexBuffer(2, CloseArrow);
   SetIndexLabel(2,"平仓价");
```

```
    return(0);
   }
//+------------------------------------------------------------------+
//| Custom indicator deinitialization function                       |
//+------------------------------------------------------------------+
int deinit()
   {
    //取消指标后，删除图表的对象
    ObjectsDeleteAll(0);
    return(0);
   }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int start()
   {
    ObjectsDeleteAll(0);
    //定义统计变量
    int BuyOrders,SellOrders,ProfitOrders;//买入、卖出订单、盈利订单计数变量
    double TotalTrades;//交易总手数
    double TotalProfit,TotalLoss;//盈利总数、亏损总数变量
    SetLable("标题栏", "老易作品  QQ:921795",200,1,8,"黑体",Red);
    SetLable("时间栏","动态报价时间:"+TimeToStr(TimeCurrent(),TIME_DATE|TIME_SECONDS)+
                      "  星期"+DayOfWeek(),4,15,9,"Verdana",Red);
    //画历史订单
    int HistoryOrderTotal=OrdersHistoryTotal()-1;//获得历史订单总数
    for (int i=1;i<=HistoryOrderTotal;i++)
       {
        OrderSelect(i,SELECT_BY_POS,MODE_HISTORY);//选择订单
        int OpenBar=iBarShift(Symbol(),0,OrderOpenTime());//获取开仓价柱数
        int CloseBar=iBarShift(Symbol(),0,OrderCloseTime());//获取平仓价柱数
        if (OrderType()==0) //买入订单统计
           {
            OpenBuyArrow[OpenBar]=OrderOpenPrice();//画买单箭头
            BuyOrders=BuyOrders+1;//买入订单数累计
            //计算盈亏总数
            if (OrderProfit()>0)
               {
                TotalProfit=TotalProfit+OrderProfit();//总盈利累计
                ProfitOrders=ProfitOrders+1;//盈利订单数累计
               }
            if (OrderProfit()<0) TotalLoss=TotalLoss+OrderProfit();//总亏损累计
           }
        if (OrderType()==1) //卖出订单统计
```

```
            {
              OpenSellArrow[OpenBar]=OrderOpenPrice();//画卖单箭头
              SellOrders=SellOrders+1;//卖出订单累计
              //计算盈亏总数
              if (OrderProfit()>0)
                 {
                  TotalProfit=TotalProfit+OrderProfit();//总盈利累计
                  ProfitOrders=ProfitOrders+1;//盈利订单数累计
                 }
              if (OrderProfit()<0) TotalLoss=TotalLoss+OrderProfit();//总亏损累计
            }
          TotalTrades=TotalTrades+OrderLots();//交易总手数
          CloseArrow[CloseBar]=OrderClosePrice();//画平仓符号

SetObj(OrderTicket(),OrderType(),OrderOpenTime(),OrderOpenPrice(),OrderCloseTime(),OrderCl
osePrice());
        }
    //画持仓订单，动态显示开仓价与当前价的连线
    int NowOrderTotal=OrdersTotal()-1;//获得持仓订单总数
    for (int j=0;j<=NowOrderTotal;j++)
        {
          OrderSelect(j,SELECT_BY_POS,MODE_TRADES);//选择订单
          //删除旧连线对象
          string NowObjectName="订单号："+DoubleToStr(OrderTicket(),0);
          ObjectDelete(NowObjectName);
          int NowOpenBar=iBarShift(Symbol(),0,OrderOpenTime());//获取开仓价柱数
          int NowCloseBar=iBarShift(Symbol(),0,OrderCloseTime());//获取平仓价柱数
          if (OrderType()==0) OpenBuyArrow[NowOpenBar]=OrderOpenPrice();//画买单箭头
          if (OrderType()==1) OpenSellArrow[NowOpenBar]=OrderOpenPrice();//画卖单箭头

SetObj(OrderTicket(),OrderType(),OrderOpenTime(),OrderOpenPrice(),Time[0],Close[0]);// 画 连
线
        }
    //显示统计信息
    SetLable("交易单统计","历史交易单总计:"+HistoryOrderTotal
                                +"   (买入订单:"+BuyOrders
                                +"   卖出订单:"+SellOrders+")"
                ,4,35,9,"Verdana",Gray);
    SetLable("  胜  率  ","  盈  利  订  单  百  分
比:"+DoubleToStr((ProfitOrders*0.01)/(HistoryOrderTotal*0.01)*100,2)+"%"
                ,4,50,9,"Verdana",Gray);
    SetLable("盈亏统计", "净盈利:"+DoubleToStr(TotalProfit+TotalLoss,2)
                                +"   (总获利:"+DoubleToStr(TotalProfit,2)
                                +"   总亏损:"+DoubleToStr(TotalLoss,2)+")"
```

```
                                ,4,65,9,"Verdana",Gray);
    SetLable("账户余额", "账户余额:"+DoubleToStr(AccountBalance(),2)
                                +"   账户净值:"+DoubleToStr(AccountEquity(),2)
                  ,4,80,9,"Verdana",Gray);
    SetLable("下单量", "总下单量(手):"+DoubleToStr(TotalTrades,2),4,95,9,"Verdana",Gray);
    return(0);
   }

/*
函数：在屏幕上画线
        myOrderTicket：订单号，myOrderType：订单类型，myOpenTime：开仓时间，
myOpenPrice：开仓价格，
        myCloseTime：平仓时间，myClosePrice：平仓价格
*/
void   SetObj(int myOrderTicket,int myOrderType,int myOpenTime,double myOpenPrice,int
myCloseTime,double myClosePrice)

    {
     string myObjectName="订单号："+DoubleToStr(myOrderTicket,0);

ObjectCreate(myObjectName,OBJ_TREND,0,myOpenTime,myOpenPrice,myCloseTime,myClosePr
ice);//画趋势线，确定两点坐标
      if (myOrderType==0) ObjectSet(myObjectName,OBJPROP_COLOR,Green);
      if (myOrderType==1) ObjectSet(myObjectName,OBJPROP_COLOR,Red);
      ObjectSet(myObjectName,OBJPROP_STYLE,STYLE_DOT);
      ObjectSet(myObjectName,OBJPROP_WIDTH, 1);
      ObjectSet(myObjectName,OBJPROP_BACK,false);
      ObjectSet(myObjectName,OBJPROP_RAY,false);
    }



/*
函数：在屏幕上显示标签
        LableName：标签名称；LableDoc：文本内容；LableX：标签 X 位置；LableY：标签 Y
位置；
        DocSize：文本字号；DocStyle：文本字体；DocColor：文本颜色
*/
void SetLable(string LableName,string LableDoc,int LableX,int LableY,
                    int DocSize,string DocStyle,color DocColor)
    {
    ObjectCreate(LableName, OBJ_LABEL, 0, 0, 0);
    ObjectSetText(LableName,LableDoc,DocSize,DocStyle,DocColor);
    ObjectSet(LableName, OBJPROP_XDISTANCE, LableX);
    ObjectSet(LableName, OBJPROP_YDISTANCE, LableY);
```

```
}
```

# 第四章　MQL4 技术指标

做金融投机的人都知道对市场的分析不外乎两个方面，一是基本面分析，二是技术分析。基本面包含了政策因素、自然环境因素以及庄家操控因素，技术面则是对市场数据进行分析。

外汇市场是一个由若干做市商自然形成的、24 小时连续交易的市场，迄今为止没有任何一个机构能够准确统计每日的成交量，只是估计每天有约 3 万亿美元的成交量。因此我们可以得出一个结论：技术分析预测在外汇市场应该成为主要的判断行情的手段。

要学习技术分析，首先必须认同技术分析的三大假设：市场行为包容消化一切；价格以趋势方式演变；历史会重演。

一、 市场行为包容消化一切

所有基本面的因素最终都会体现在市场行为上。外汇市场如此之大，以至于任何国家货币政策和“庄家”的力量都显得微不足道，这个市场更能充分体现市场的供求关系。

二、 价格以趋势方式演变

价格根据供求关系的变化而变化，因此是有规律可循的。市场上过度的买入和卖出都会导致价格的回调，我们通过一些技术手段就能分析预测市场的发展趋势，以此指导我们炒汇行为。

三、 历史会重演

市场不仅仅反映供求关系，还反映了人们的心理。在图表分析过程中我们能发现价格波动存在着与过去太多的惊人相似之处。

我不承认自己是个“技术派”，但在面对看似纷繁复杂、实则简单易懂的外汇市场时，我觉得技术分析是我们炒汇的最好手段。

技术分析不是万能的，没有一个技术指标能确保你赚钱，我们通常会将几个指标组合起来判断价格趋势，决定买入卖出平仓止损。

MT4 是一个被广泛使用的外汇交易平台，内含 4 大类 29 种常用技术指标。我认为我们只需要了解这些指标就足够了，如果你精力过剩，也可以研究网上近千种技术指标。

成交量类指标受外汇市场特点影响，判断趋势的效率不高。趋势类指标和震荡类指标都是根据市场价格形成的，能比较准确反映市场趋势，预测平仓止损价位。比尔威廉类指标更接近图形形态的分析，是一组非常实用的指标。

“一目平衡表”指标又叫“日本云”，是一个非常有意思的指标，它居然能预测未来若干个蜡烛的趋势，我不得不佩服日本人的聪明。而蜡烛图也是日本人发明的，形象的表述了一个时间周期的市场变动过程，大有八卦周易的涵义，很值得我们细细品味。

“移动平均线”是所有接触过股票期货的人都常用的指标，但又有多少人理解其精髓呢？一根 MA 曲线能体现市场趋势，两根 MA 曲线能提示市场反转信号，三根 MA 曲线可以确定一段趋势的长度。不同的货币对，不同的时间周期，不同的平均方法，不同的曲线组合，都会带来不同的判断结果，这都需要我们认真比对。

“平均真实范围指标”居然能测算出止损价格，恐怕谁都会将信将疑。

技术分析中经常会使用“神奇数字”，这确实是一组神奇数字。这组数字是“1，2，3，5，8，13，21，34，55……”，你能找到其中的规律么？这组数字在技术分析中又怎么使用呢？

技术分析不仅仅能给你带来好的收益，更能给你带来探索未知的乐趣。

MT4 平台集成了 30 个默认技术指标，指标按照类型分为成交量指标、趋势指标、震荡

指标和比尔威廉指标四类。

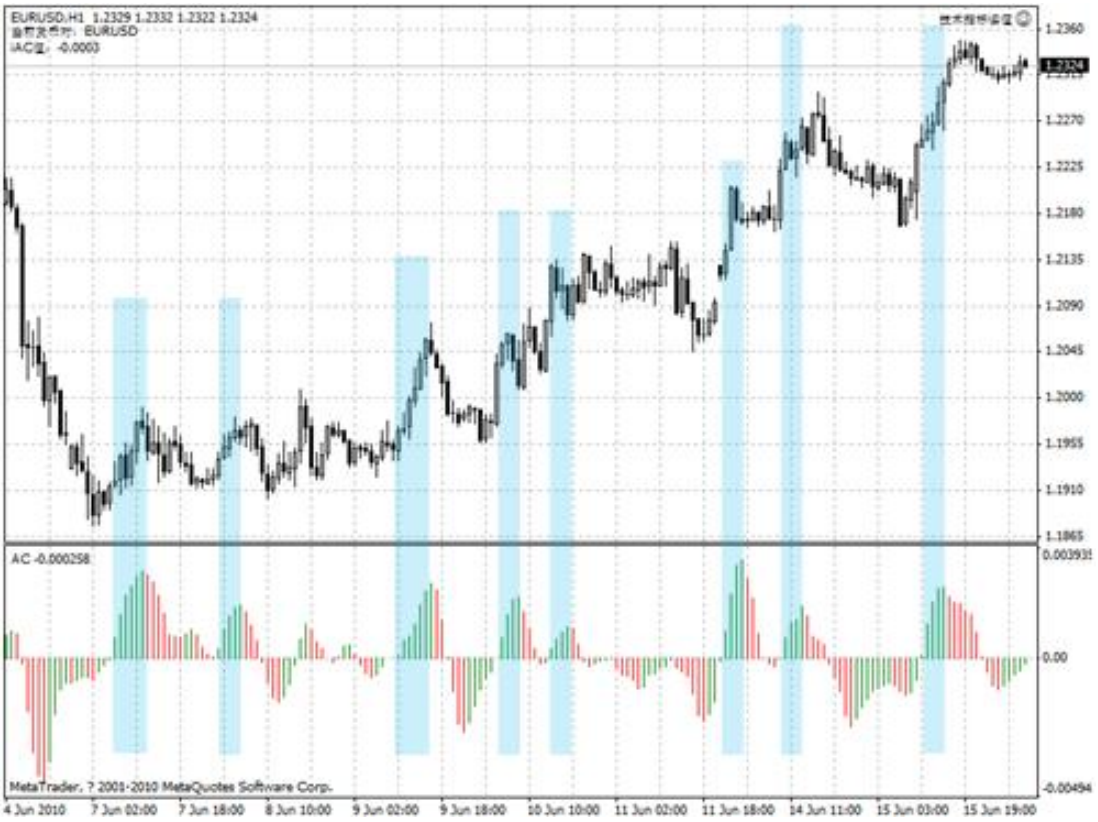
下表总结了 30 个指标的基本特性，推荐指数星数越多的说明实用性越好。

| 类型 | 指标名称 | | 指标函数 | 适用周期 | 推荐指数 | 联动指标 |
|---|---|---|---|---|---|---|
| | 指标中文名 | 指标英文名 | | | | |
| 成交量指标 | 资金流量指数指标 | Money Flow Index | iMFI | 所有 | ★★★ | RSI |
| | 能量潮指标 | On Balance Volume | iOBV | 所有 | | |
| | 离散指标 | Accumulation / Distribution | iAD | 所有 | | |
| 趋势指标 | 移动平均线指标 | Moving Average | iMA | 所有 | ★★★★★ | |
| | 抛物线状止损和反转指标 | Parabolic SAR | iSAR | 所有 | ★★ | |
| | 标准离差指标 | Standard Deviation | iStdDev | | | |
| | 平均方向移动指标 | Average Directional Movement Index | iADX | H1+ | ★★★★ | |
| | 保力加通道技术指标 | Bollinger Band | iBands | 所有 | ★★★★★ | |
| | 商品通道指标 | Commodity Channel Index | iCCI | H1- | ★★★★ | |
| 震荡指标 | 移动平均震荡指标 | Moving Average of Osillator | iOsMA | 所有 | | MACD |
| | 相对强弱指标 | Relative Strength Index | iRSI | 所有 | ★★★★★ | |
| | 相对活力指数指标 | Relative Vigor Index | iRVI | 所有 | | |
| | 随机震荡指标 | Stochastic Oscillator | iStoch | 所有 | ★★★★ | |
| | 平均真实范围指标 | Average True Range | iATR | H1+ | ★★★ | |
| | 熊力震荡指标 | Bears Power | iBearsPower | 所有 | | |
| | 牛力震荡指标 | Bulls Power | iBullsPower | 所有 | | |
| | | DeMarker | iDeMarker | 所有 | ★★★ | |
| | 包络指标 | Envelops | iEnvelopes | H1- | ★★★★ | |
| | 强力指标 | Force Index | iForce | 所有 | | MA |
| | 一目平衡表指标 | Ichimoku Kinko Hyo | iIchimoku | D1+ | ★★★★ | |
| | 移动平均汇总/分离指标 | MACD | iMACD | 所有 | ★★★★★ | |
| | 动量索引指标 | Momentum | iMomentum | 所有 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 威廉指标 | Williams’ Percent Range | iWPR | 所有 | ★★★★ | |
| 比尔威廉指标 | 震荡加速指标 | Accelerator Oscillator | iAC | 所有 | | |
| | 鳄鱼指标 | Alligator | iAlligator | 所有 | ★★★★★ | |
| | 振荡指标 | Awesome Oscillator | iAO | D1+ | ★★★ | |
| | 分形指标 | Fractals | iFractals | 所有 | ★★★★ | Alligator |
| | 加多摆动指标 | Gator Oscillator | iGator | 所有 | | |
| | 市场促进指数指标 | Market Facilitation Index | iBWMFI | 所有 | | |

# 4.1 Accelerator Oscillator 震荡加速指标

iAC 属于比尔威廉指标，反映当前趋势的加速和减速，该值大于前值则用绿色表示，小于前值用红色表示。




【用法】

1、 iAC 值大于 0 递增，市场处于上涨阶段；

2、 iAC 值大于 0 递减，市场处于盘整回调阶段；

3、 iAC 值小于 0 递减，市场处于下跌阶段；

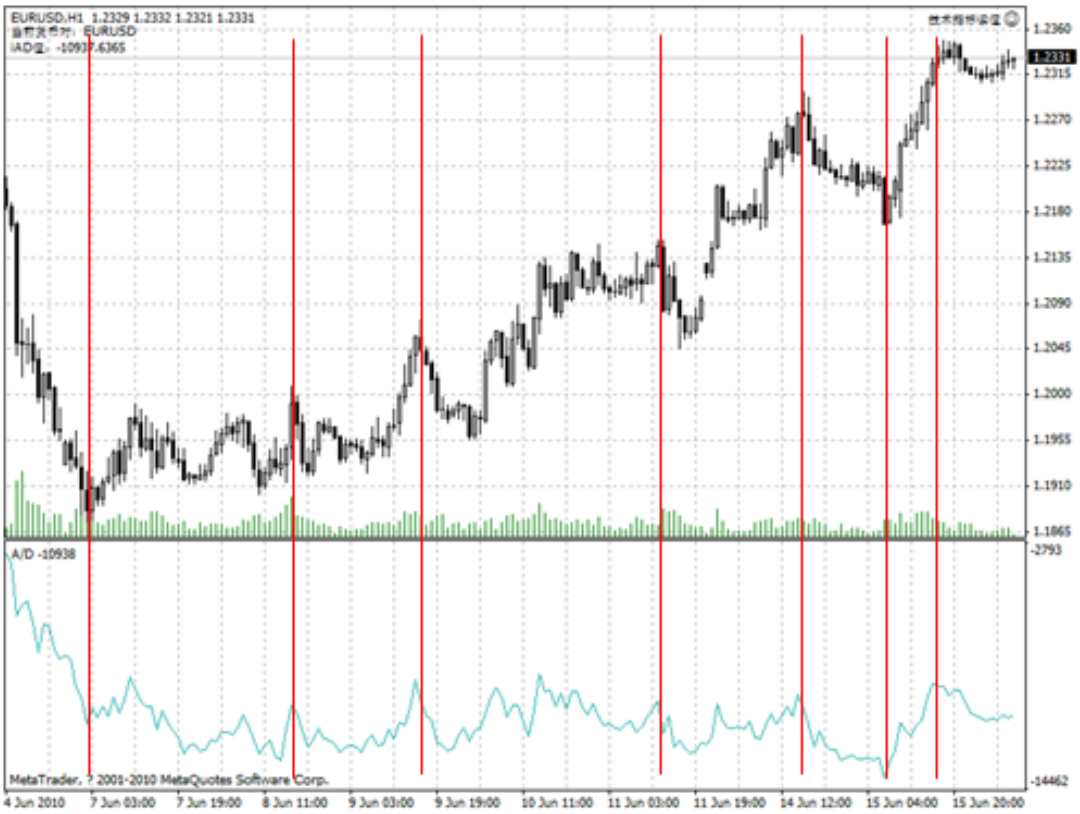
4、 iAC 值小于 0 递增，市场处于盘整回调阶段。

【语法】double iAC(string symbol, int timeframe, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

## 4.2 Accumulation/Distribution 离散指标

iAD 属于成交量指标，是由价格和成交量计算出来，市场在一轮单边行情当中会逐步积累很多的同方向订单，积累到一定程度市场将出现反转。




由于外汇交易是若干做市商组成的，成交量实际上是个不确定的因素，因此该项指标在外汇分析中不是一个重要的指标，这个指标读数没有确定的范围，很难精确把握。

【用法】

图中红线表示 AD 指标破位后出现的新的一轮行情，AD 指标有一定的超前性。需要配合画趋势线来判断破位。

【语法】double iAD(string symbol, int timeframe, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

# 4.3 Alligator 鳄鱼指标

iAlligator 属于比尔威廉指标，根据中线价格（最高最低价的中间价）形成 3 条曲线，由于形状像鳄鱼嘴巴，被外国人极有想象力的命名为“鳄鱼指标”。


【用法】

1、 周期参数选择 13、8、5，这是一组“神奇数字”；

2、 相对偏移量选择 8、5、3，可钝化市场趋势，用损失部分行情为代价，换取震荡行情可能带来的损失；

3、 绿线>红线>蓝线，市场处于上涨阶段；

4、 绿线<红线<蓝线，市场处于下跌阶段；

5、 绿线、红线、蓝线没有顺序，市场处于盘整阶段。

【语法】double iAlligator( string symbol, int timeframe, int jaw_period, int jaw_shift, int teeth_period, int teeth_shift, int lips_period, int lips_shift, int ma_method, int applied_price, int mode, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | jaw_period | 鳄鱼下颚平均周期，蓝线。默认选 13 |
| 4、 | jaw_shift | 蓝线相对偏移量。默认选 8 |
| 5、 | teeth_period | 鳄鱼牙齿平均周期，红线。默认选 8 |
| 6、 | teeth_shift | 红线相对偏移量。默认选 5 |
| 7、 | lips_period | 鳄鱼嘴唇平均周期，绿线。默认选 5 |
| 8、 | lips_shift | 绿线相对偏移量。默认选 3 |

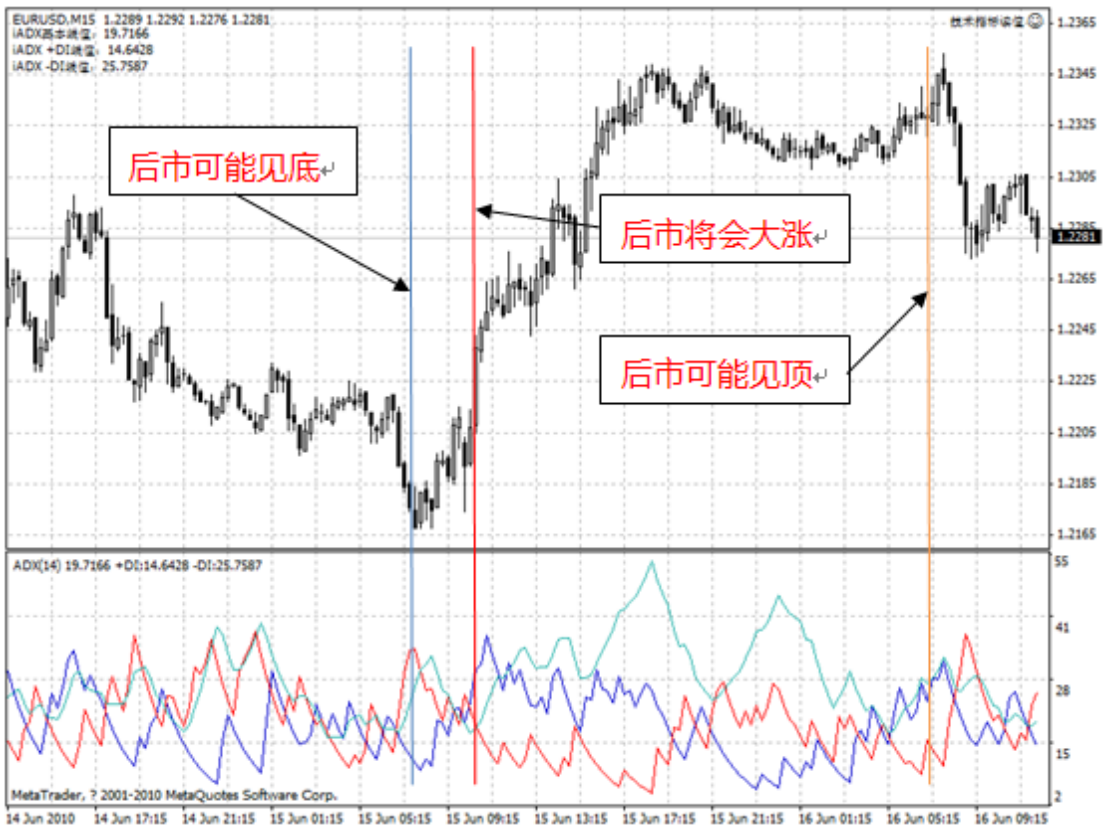
9、 ma_method MA 方法。默认取指数平均 MODE_EMA
10、 applied_price 应用价格。默认取中线价 PRICE_MEDIAN
11、 mode 返回数据，MODE_GATORJAW 为下颚，MODE_GATORTEETH 为牙齿，MODE_GATORLIPS 为嘴唇
12、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

```
iAlligator(NULL,0,13,8,8,5,5,3,MODE_EMA,PRICE_MEDIAN,MODE_GATORJAW,0)
iAlligator(NULL,0,13,8,8,5,5,3,MODE_EMA,PRICE_MEDIAN,MODE_GATORTEETH,0)
iAlligator(NULL,0,13,8,8,5,5,3,MODE_EMA,PRICE_MEDIAN,MODE_GATORLIPS,0)
```

## 4.4 Average Directional Movement Index 平均方向移动指标

iADX 属于趋势指标，适合中长线预测，能够比较准确的认定市场行情。




【用法】蓝线为+DI，红线为-DI，青线为 ADX 基本线（周期 14）

1、+DI 上穿-DI 和 ADX，同时 ADX 跟涨，市场将进入大涨阶段；
2、ADX 一般在 20~40 之间，超过 25 以上，市场上涨阶段开始；
3、+DI 与-DI 经常交叉，且 ADX 在 20 以下，市场进入盘整阶段；
4、+DI 在-DI 之上，且差距大，同时 ADX 升破这两条线，有回落迹象，说明市场即将见顶；
5、-DI 在+DI 之上，且差距大，同时 ADX 升破这两条线，有回落迹象，说明市场即将见底；
6、ADX 读数偏高，市场进入超买超卖阶段；
7、ADX 低于 25，市场进入盘整阶段。

【语法】double iADX( string symbol, int timeframe, int period, int applied_price, int mode, int shift)

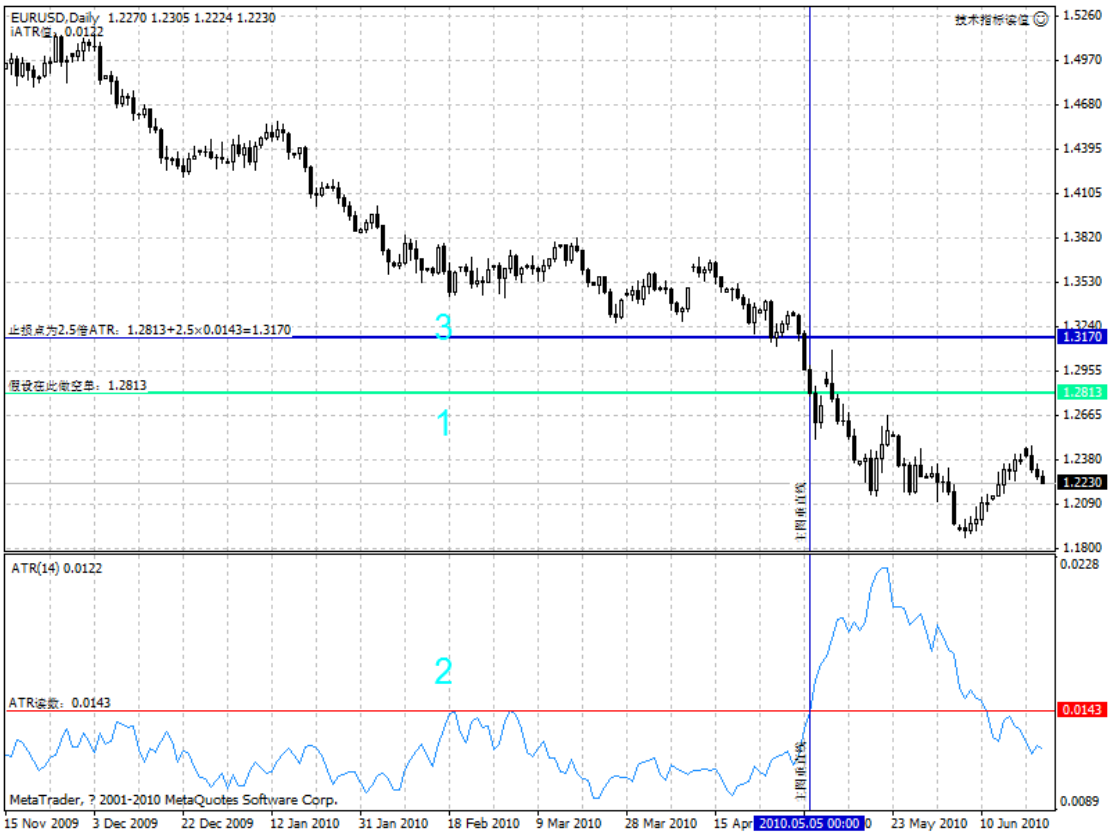
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 计算平均周期。默认选 14 |
| 4、 | applied_price | 应用价格。默认取平仓价 PRICE_CLOSE |
| 5、 | mode | 返回数据， MODE_MAIN 为基本指标线，MODE_PLUSDI 为+DI 指标，MOSE_MINUSDI 为-DI 指标线 |
| 6、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iADX(NULL,0,14,PRICE_CLOSE,MODE_MAIN,0)
iADX(NULL,0,14,PRICE_CLOSE,MODE_PLUSDI,0)
iADX(NULL,0,14,PRICE_CLOSE,MODE_MINUSDI,0)
```

# 4.5 Average True Range 平均真实范围指标

iATR 属于震荡指标，反映市场震荡范围，用于确定止损价位。




【用法】

1、 ATR 读数是这个指标可能震荡的范围，如图当前读数 0.0122，说明当前货币对（EURUSD）在当前时间周期（Daily）中价格震荡范围为 0.0122。不同的货币对、不同的时间周期读数不一样；

2、 ATR 读数越高说明价格波动范围越大，读数越低价格波动范围越小；

3、 ATR 适合计算止损价格。如图在 1 号线 1.2813 做空，ATR 读数为 2 号线 0.0143，止损范围设置为 2.5 倍 ATR 即 0.0143×2.5=0.0357，止损价 3 号线为 1.2813+0.0357=1.3170；

4、 ATR 指标中长期止损范围通常为 2.5~4 倍 ATR 之间，目的为了过滤掉市场震荡因素。

【语法】double iATR( string symbol, int timeframe, int period, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

3、 period 计算平均周期。默认选 14

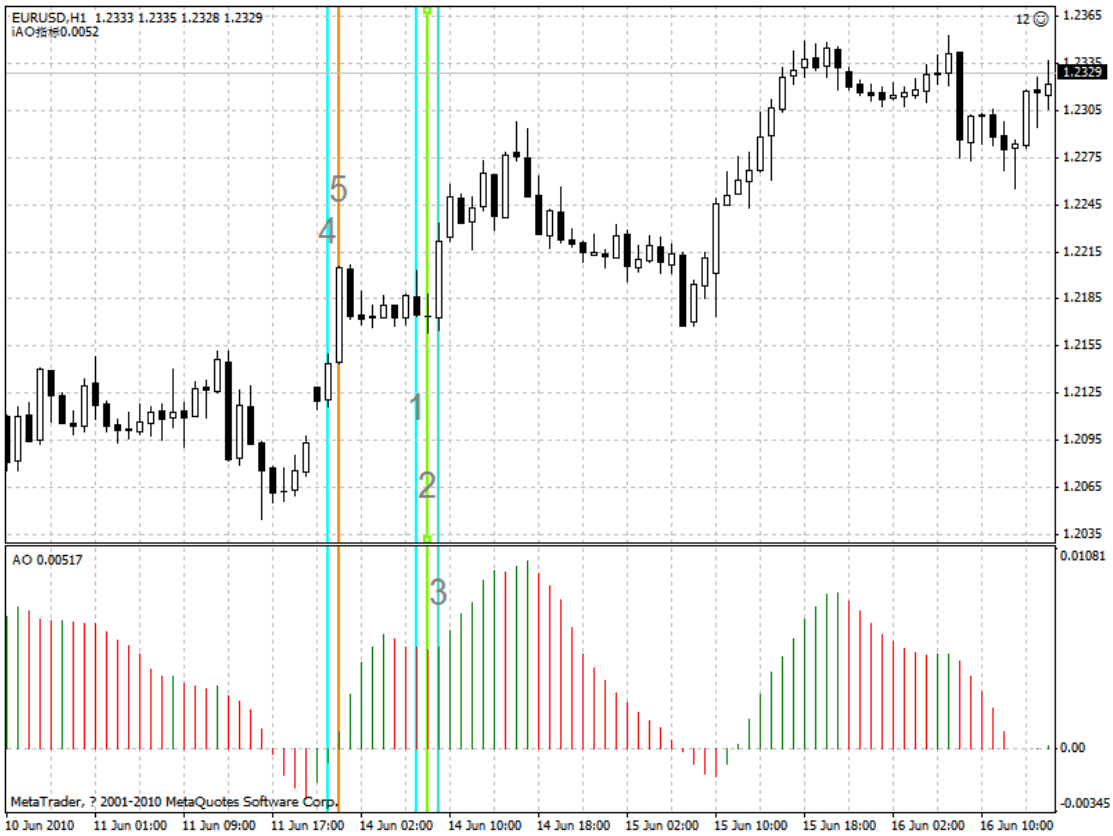
4、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

```
iATR(NULL,0,14,0)
```

## 4.6 Awesome Oscillator 振荡指标

iAO 属于比尔威廉指标，又叫做动量震荡指标，提供买入卖出信号。




【用法】

1、 AO 指标是一个中长线指标，建议与 AC 同时使用；

2、 AO 值大于 0 为买方市场；

3、 AO 三线买入信号：如图当 2 号线小于 1 号线和 3 号线时，指标发出买入信号；

4、 AO 两线买入信号（零线买入）：如图 4 号线小于 0，5 号线大于 0，指标发出买入信号；

5、 卖出判断与上面所述相反。

【语法】double iAO( string symbol, int timeframe, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

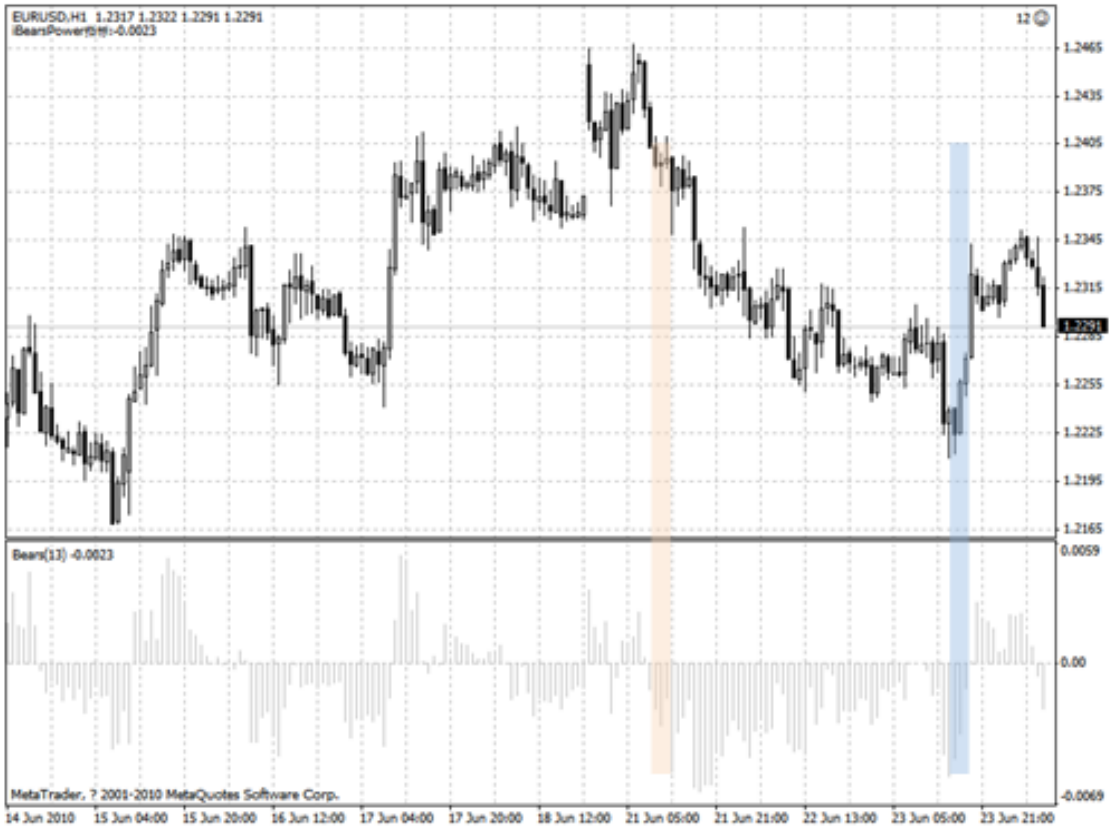
3、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

iAO(NULL,0,0)

## 4.7 Bears Power 熊力震荡指标

iBearsPower 属于震荡指标，提供市场买入信号。




【用法】

1、 BearsPower 为负数，同时逐渐增大，表示市场出现了买入信号；

2、 BearsPower 为负数，同时逐渐减小，表示市场出现了卖出信号；

3、 该指标通常与牛力震荡指标联合使用。

【语法】double iBearsPower( string symbol, int timeframe, int period, int applied_price, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

3、 period 计算平均周期。默认选 13

4、 applied_price 选择价格，默认选收盘价 PRICE_CLOSE

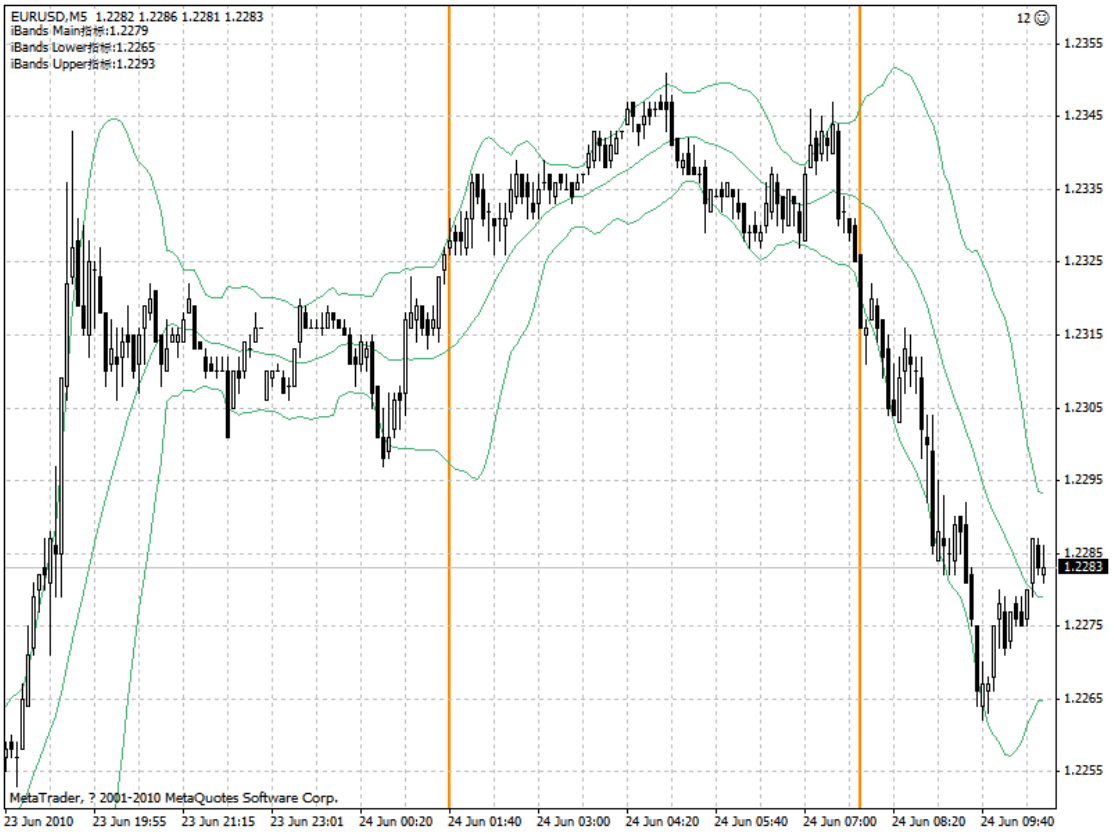
5、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

iBearsPower(NULL,0,13,PRICE_CLOSE,0)

# 4.8 Bollinger Bands 保力加通道技术指标

iBands 属于趋势指标，判断市场运动趋势的指标，用来确定支撑位、阻力位、反转信号等。




【用法】

1、 价格突破 Bands 上线时，预示着涨势的开始；

2、 价格突破 Bands 下线时，预示着跌势的开始；

3、 价格回归到上下线之间，且突破中心线，预示市场趋势不明朗。

【语法】double iBands( string symbol, int timeframe, int period, int deviation, int bands_shift, int applied_price, int mode, int shift)

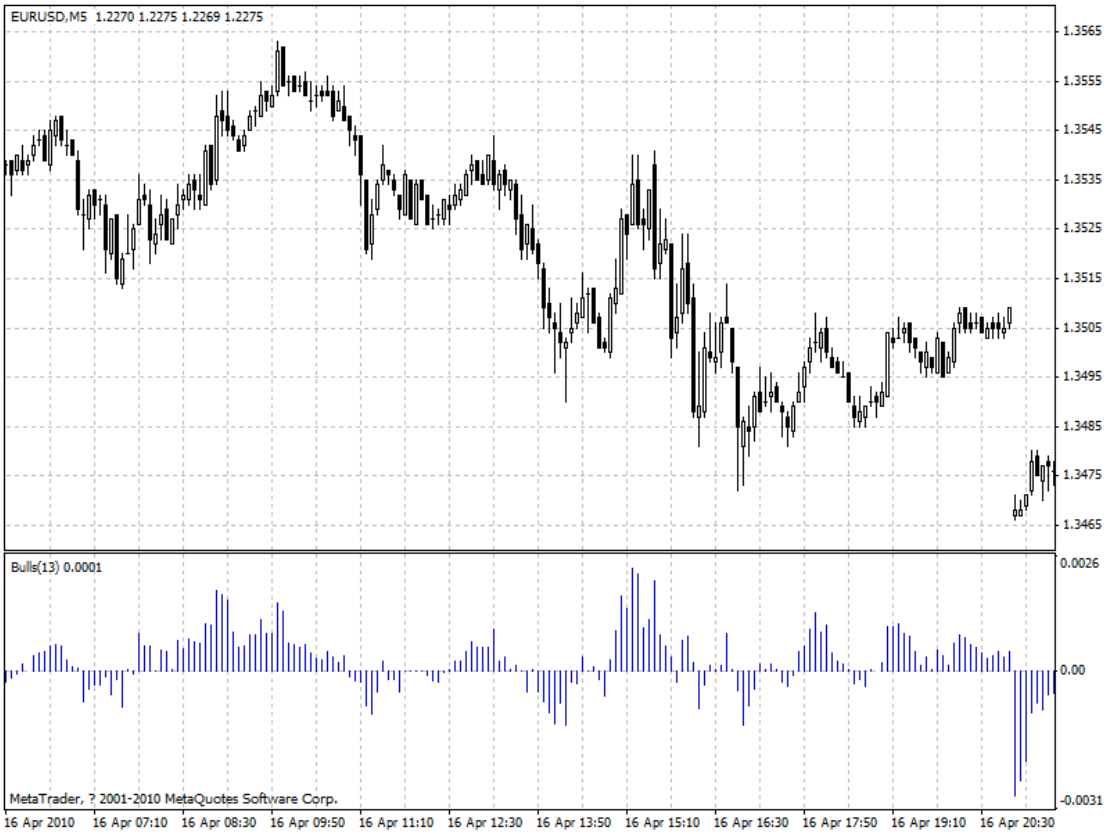
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 计算平均周期。默认选 20 |
| 4、 | deviation | 与主线偏差。默认选 2 |
| 5、 | bands_shift | 平移量。默认选 0 |
| 6、 | applied_price | 应用价格。默认取最低价 PRICE_CLOSE |
| 7、 | mode | 返回读数， MODE_UPPER 为上面线，MODE_LOWER 为下面线，MODE_MAIN 为中间线 |
| 8、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iBands(NULL,0,20,2,0,PRICE_CLOSE,MODE_MAIN,0)
iBands(NULL,0,20,2,0,PRICE_CLOSE,MODE_LOWER,0)
iBands(NULL,0,20,2,0,PRICE_CLOSE,MODE_UPPER,0)
```

## 4.9 Bulls Power 牛力震荡指标

iBullsPower 属于震荡指标，提供市场卖出信号。




【用法】

1、 BullsPower 为正数，同时逐渐增大，表示市场出现了买入信号；

2、 BullsPower 为正数，同时逐渐减小，表示市场出现了卖出信号；

3、 该指标通常与熊力震荡指标联合使用。

【语法】double iBullsPower( string symbol, int timeframe, int period, int applied_price, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

3、 period 计算平均周期。默认选 13

4、 applied_price 选择价格，默认选收盘价 PRICE_CLOSE

5、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

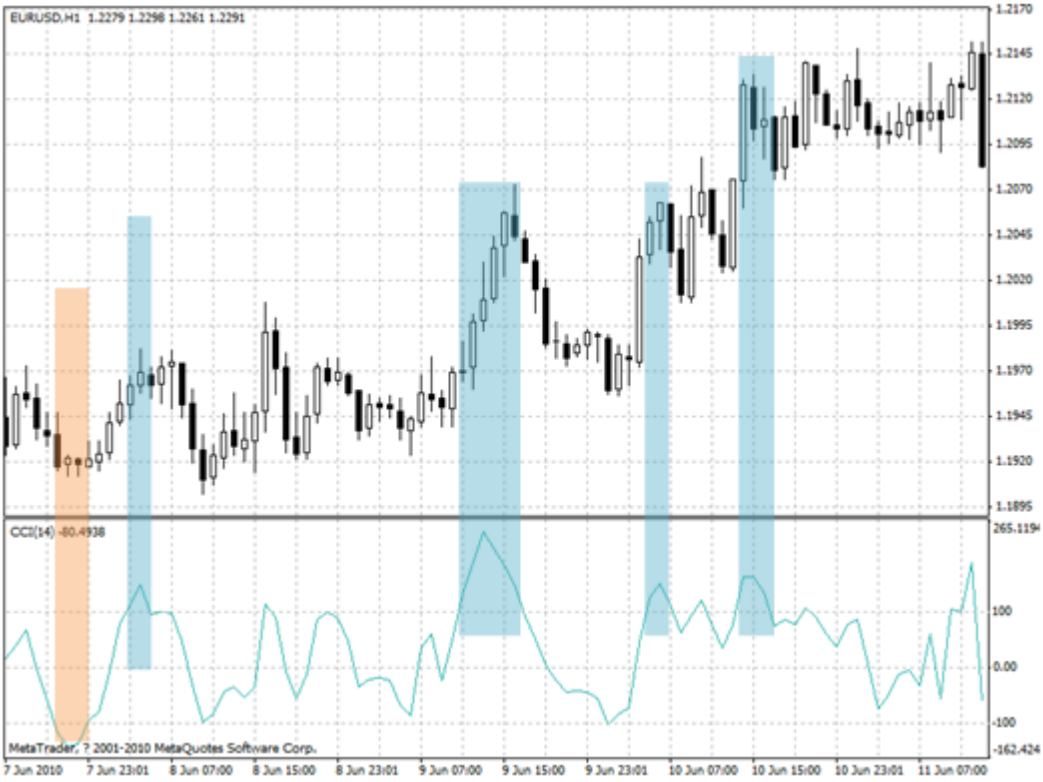
【代码】

```
iBullsPower(NULL,0,13,PRICE_CLOSE,0)
```

## 4.10 Commodity Channel Index 商品通道指标

iCCI 属于趋势指标，又叫顺势指标，能较准确的标示市场的超买超卖，适合短线操作。

【用法】




1、 CCI 读数在-100~+100 之间，市场处于震荡阶段；

2、 CCI 读数小于-100，市场处于超卖阶段；

3、 CCI 读数大于+100，市场处于超买阶段

【语法】double iCCI( string symbol, int timeframe, int period, int applied_price, int shift)

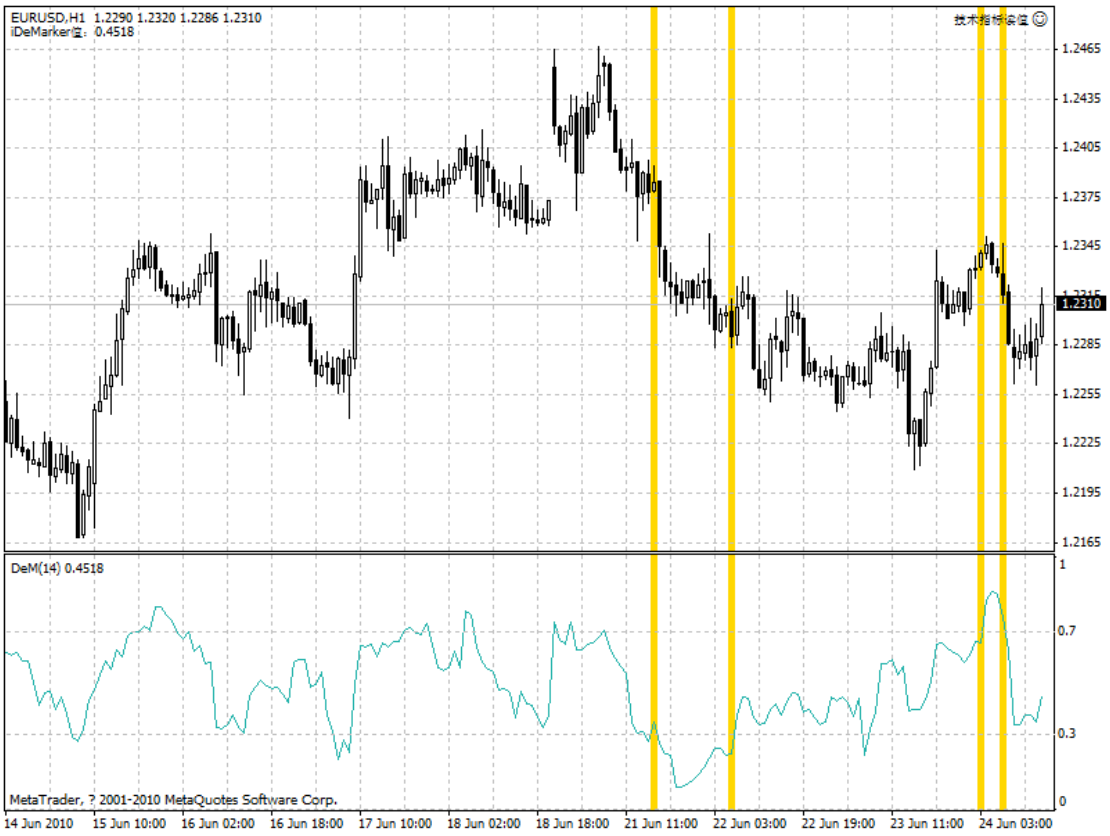
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 计算平均周期。默认选 14 |
| 4、 | applied_price | 选择价格，默认选典型价 PRICE_TYPICAL |
| 5、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iCCI(NULL,0,14,PRICE_TYPICAL,0)
```

## 4.11 DeMarker

iDeMarker 是一个震荡指标，用来预测牛市、熊市。




【用法】

1、 DeMarker 读数低于 0.3，后市可能上涨；

2、 DeMarker 读数高于 0.7，后市可能下跌。

【语法】double iDeMarker( string symbol, int timeframe, int period, int shift)

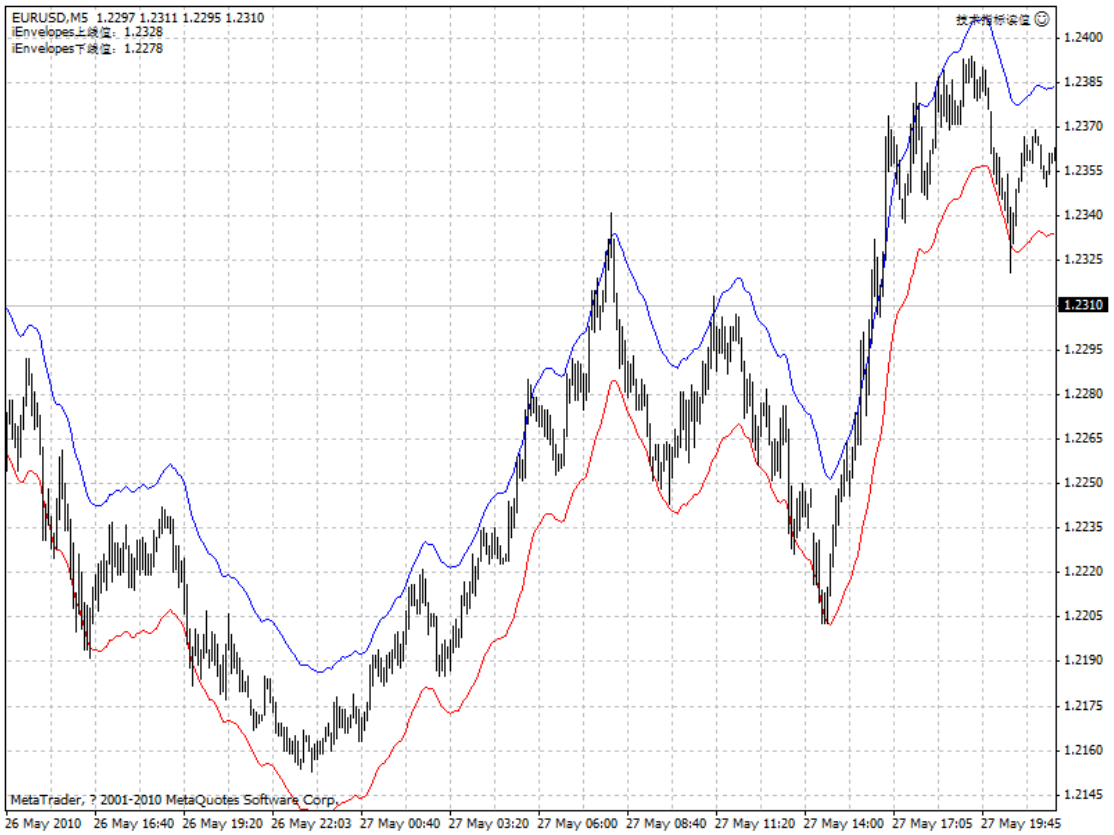
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 计算平均周期。默认选 14 |
| 4、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iDeMarker(NULL, 0, 13, 0)
```

## 4.12 Envelops 包络指标

iEnvelopes 是短线震荡指标，提示市场买入卖出信号。




【用法】

1、 价格突破 Envelopes 上线，市场将下跌；

2、 价格突破 Envelopes 下线，市场将上涨；

3、 不同货币对、不同时间周期需要调整偏差量。

【语法】double iEnvelopes( string symbol, int timeframe, int ma_period, int ma_method, int ma_shift, int applied_price, double deviation, int mode, int shift)

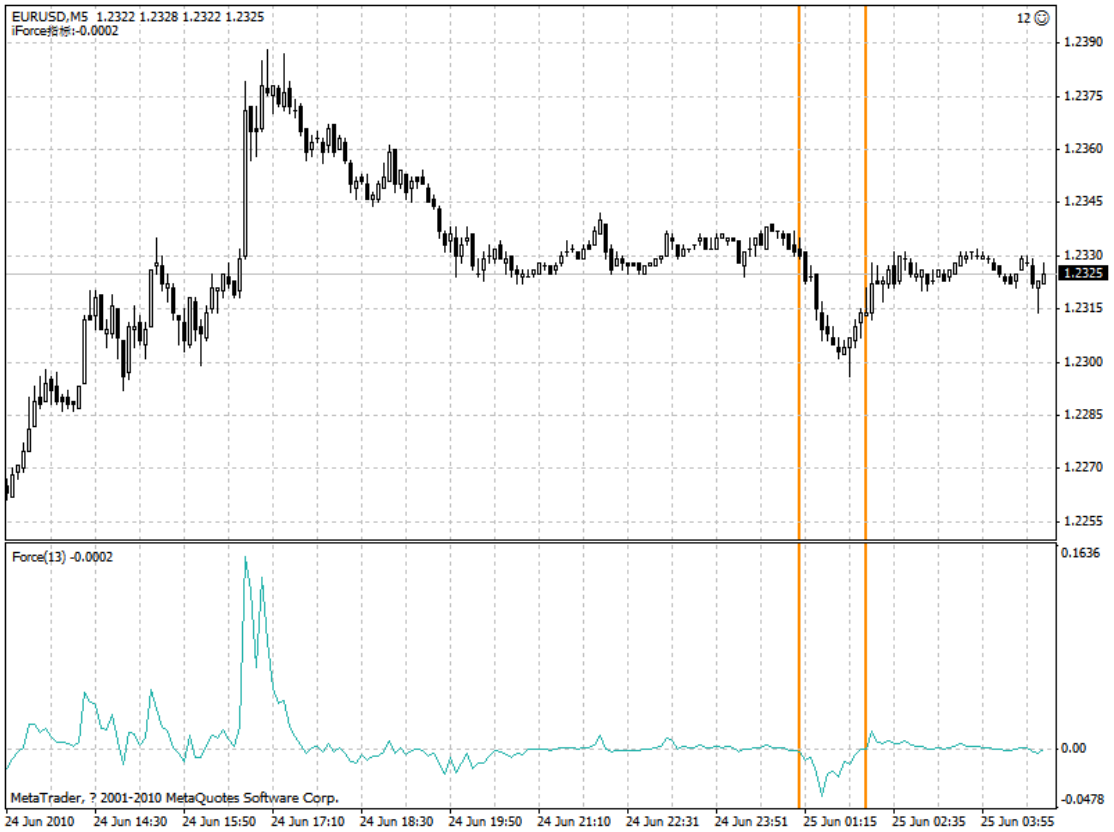
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | ma_period | 主线平均周期。默认选 13 |
| 4、 | ma_method | MA 方法，通常选指数平滑 MODE_EMA |
| 5、 | ma_shift | MA 偏移，默认选 0 |
| 6、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 7、 | deviation | 与主线偏差。根据货币对和时间周期选择，这里为 0.2 |
| 5、 | bands_shift | 平移量。默认选 0 |
| 6、 | applied_price | 应用价格。默认取最低价 PRICE_CLOSE |
| 7、 | mode | 返回读数， MODE_UPPER 为上面线，MODE_LOWER 为下面线 |
| 8、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iEnvelopes(NULL, 0, 13,MODE_EMA,0,PRICE_CLOSE,0.2,MODE_UPPER,0)
iEnvelopes(NULL, 0, 13,MODE_EMA,0,PRICE_CLOSE,0.2,MODE_LOWER,0)
```

## 4.13 Force Index 强力指标

iForce 是震荡指标，判断市场涨跌。




【用法】

1、 Force 度数小于 0，且价格下跌，市场处于跌势；

2、 Force 度数大于 0，且价格上涨，市场处于涨势；

3、 Force 指标需要与平均移动趋势指标联合使用。

【语法】double iForce( string symbol, int timeframe, int period, int ma_method, int applied_price, int shift)

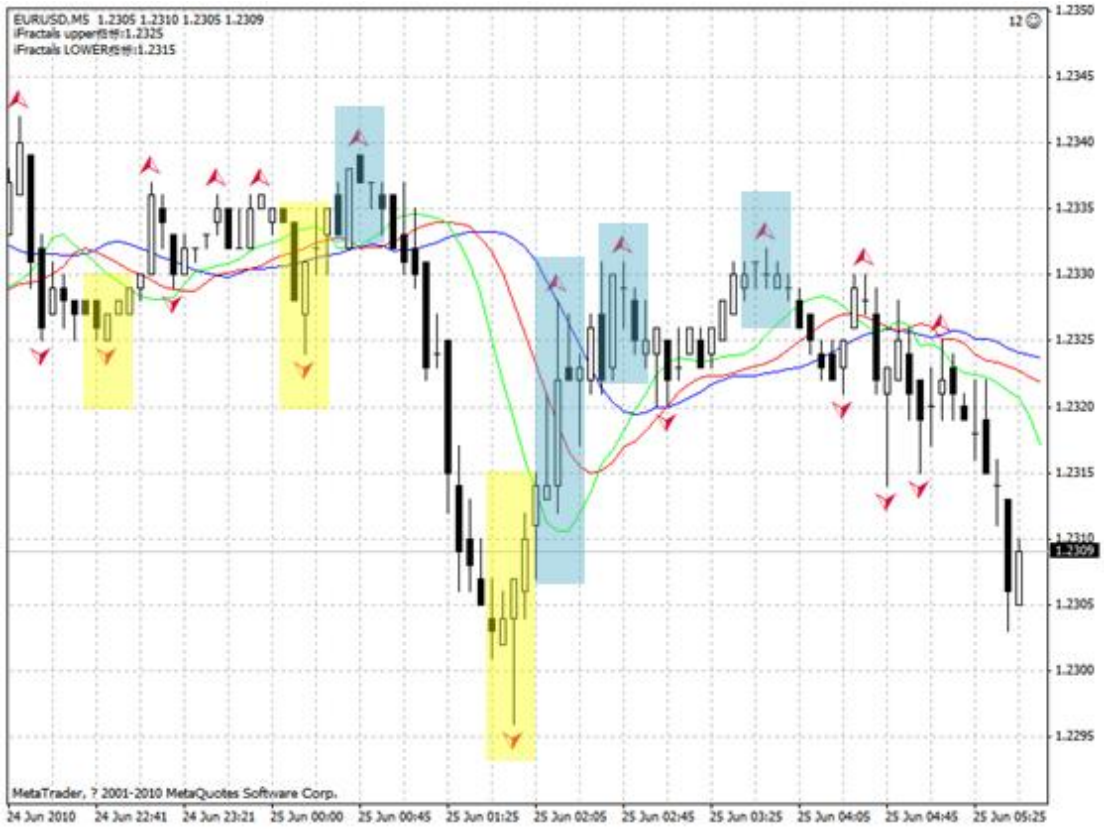
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | ma_period | 主线平均周期。默认选 13 |
| 4、 | ma_method | MA 方法，通常选指数平滑 MODE_EMA |
| 5、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 6、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iForce(NULL, 0, 13,MODE_EMA,PRICE_CLOSE,0)
```

## 4.14 Fractals 分形指标

iFractals 属于比尔威廉指标之一，与鳄鱼指标联合判断多空交易。




【用法】

1、 程序方法读出前面的 Fractals 值；

2、 5 个连续的蜡烛图，如果出现中间的最高价大于两边的最高价，则形成上分形箭头，反之形成下分形箭头，如图蓝色标注为上分形，黄色标注为下分形；

3、 Fractals 值小于牙齿（红线），做多，或者多仓继续持有；

4、 Fractals 值大于牙齿，做空，或者空仓继续持有；

5、 5 个连续蜡烛图内出现 2 个以上分形箭头，停止交易；

6、 针对指定的货币对和时间周期，需要调整鳄鱼指标参数。

【语法】double iFractals( string symbol, int timeframe, int mode, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

3、 mode 返回读数， MODE_UPPER 为上箭头，MODE_LOWER 为下箭头

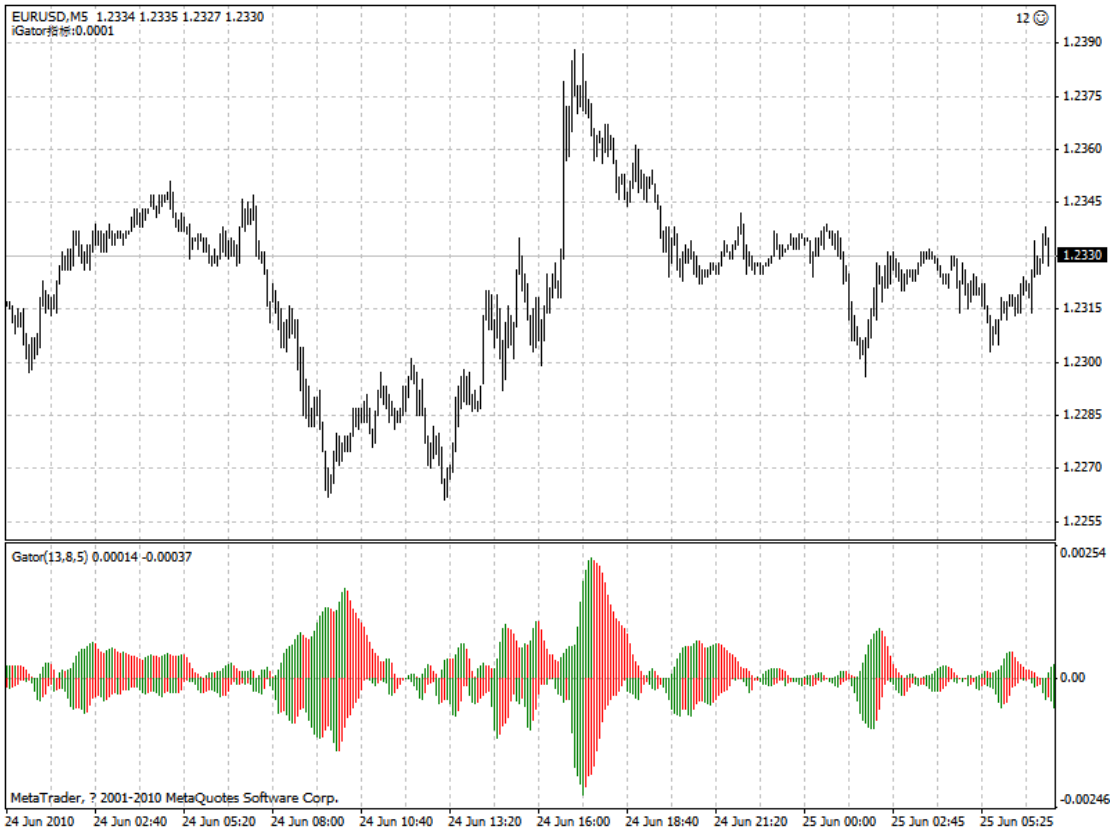
4、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

```
iFractals(NULL, 0, MODE_UPPER, 7)
iFractals(NULL, 0, MODE_LOWER, 9)
```

## 4.15 Gator Oscillator 加多摆动指标

iGator 属于比尔威廉指标之一，鳄鱼指标的变种。




【用法】很少使用

【语法】double iGator( string symbol, int timeframe, int jaw_period, int jaw_shift, int teeth_period, int teeth_shift, int lips_period, int lips_shift, int ma_method, int applied_price, int mode, int shift)

【代码】

iGator(NULL, 0, 13, 8, 8, 5, 5, 3, MODE_SMMA, PRICE_MEDIAN, MODE_UPPER, 0)

## 4.16 Ichimoku Kinko Hyo 一目平衡表指标

iIchimoku 属于震荡指标，又叫“日本云”，该发出买入卖出信号，日线或周线最有效。


【用法】

1、 转折线表示市场趋势，该线上升或者下降表示市场存在涨或跌的趋势，该线走平，说明市场进入盘整；

2、 基准线从下往上走，产生了买入信号，反之产生卖出信号；

3、 关于“云”，价格在云之上，那么云形成了两道支撑价位。价格在云之下，云形成了两道压制价位；

4、 绿线不知道是什么；

5、 基准线设置为 26 时，“云”会有 26 个预期图形，不知道该怎么看。

【语法】double iIchimoku( string symbol, int timeframe, int tenkan_sen, int kijun_sen, int senkou_span_b, int mode, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | tenkan_sen | 转折线周期，9 |
| 4、 | kijun_sen | 基准线周期，26 |
| 5、 | senkou_span_b | 延展线周期，52 |
| 6、 | mode | 返回读数， 转折线 MODE_TENKANSEN，基准线 MODE_KIJUNSEN，云线 A MODE_SENKOUSPANA，云线 B MODE_SENKOUSPANB，通道 MODE_CHINKOUSPAN |
| 7、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

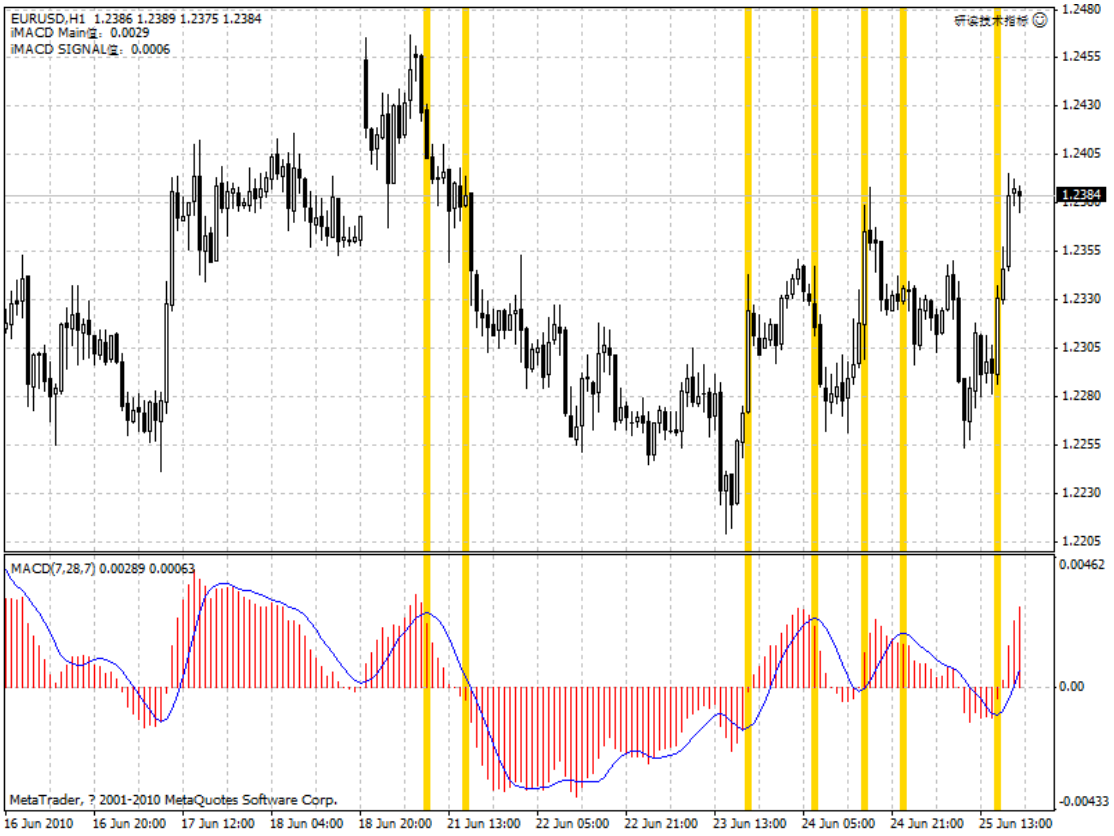
【代码】

iIchimoku(NULL, 0, 9, 26, 52, MODE_TENKANSEN, 0)

## 4.17 MACD 移动平均汇总/分离指标

iMACD 属于震荡指标，用来确定超买超卖信号。

【用法】




1、 如图，MACD 双线交叉，后市将有反转；

2、 MACD 信号线（蓝线）上穿 0 线，且与主线（红线）背离，市场出现涨势；

3、 MACD 信号线（蓝线）下穿 0 线，且与主线（红线）背离，市场出现跌势；

4、 MACD 信号线与主线读值同为正数，且差距不大，市场处于盘整阶段；

5、 MACD 与 ADX 结合在 H1 操作中胜率非常高（参见 http://wudi20052007.blog.sohu.com/141315188.html）。

【语法】double iMACD( string symbol, int timeframe, int fast_ema_period, int slow_ema_period, int signal_period, int applied_price, int mode, int shift)

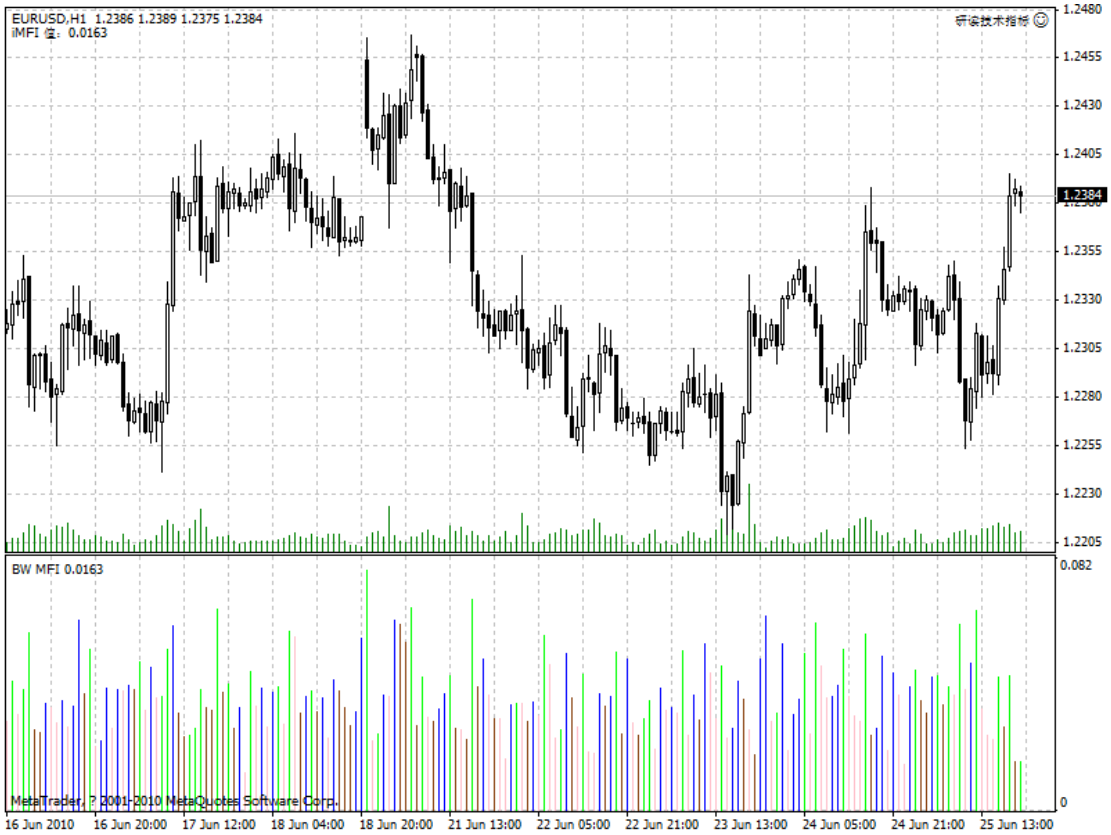
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | fast_ema_period | 快速线周期 |
| 4、 | slow_ema_period | 慢速线周期 |
| 5、 | signal_period | 信号线周期 |
| 6、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 7、 | mode | 返回读数，主线 MODE_MAIN，信号线 MODE_SIGNAL |
| 8、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

iMACD(NULL,0,7,28,7,PRICE_CLOSE,MODE_MAIN,0)
iMACD(NULL,0,7,28,7,PRICE_CLOSE,MODE_SIGNAL,0)

## 4.18 Market Facilitation Index 市场促进指数指标

iBWMFI 是一个比尔威廉指标，显示市场价格与成交量的一种状态。




【用法】

1、 该指标与交易量相关，意义不大；

2、 这是一个判断图形形态的指标。

【语法】double iBWMFI( string symbol, int timeframe, int shift)

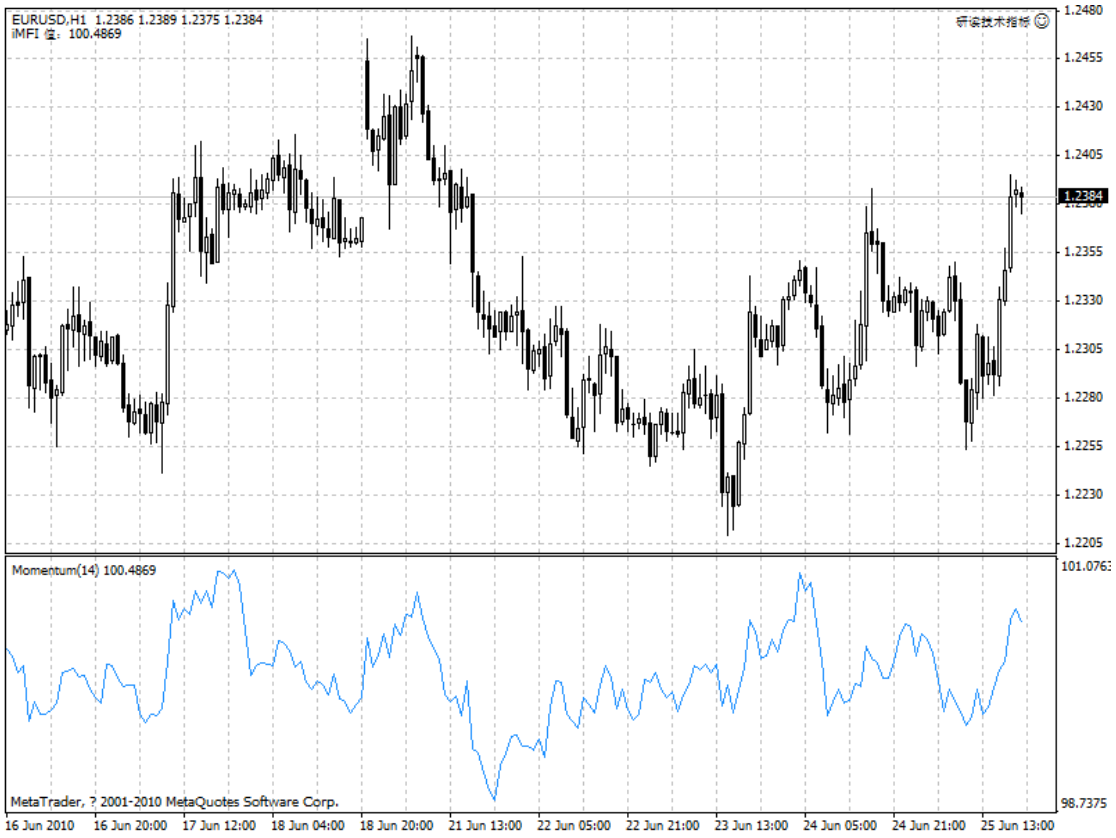
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

iBWMFI(NULL, 0, 0)

## 4.19 Momentum 动量索引指标

iMomentum 属于震荡指标，用来预测价格涨跌。




【用法】

1、 Momentum 指标需要与其他指标联合使用；

2、 不清楚 Momentum 读值范围。

【语法】double iMomentum( string symbol, int timeframe, int period, int applied_price, int shift)

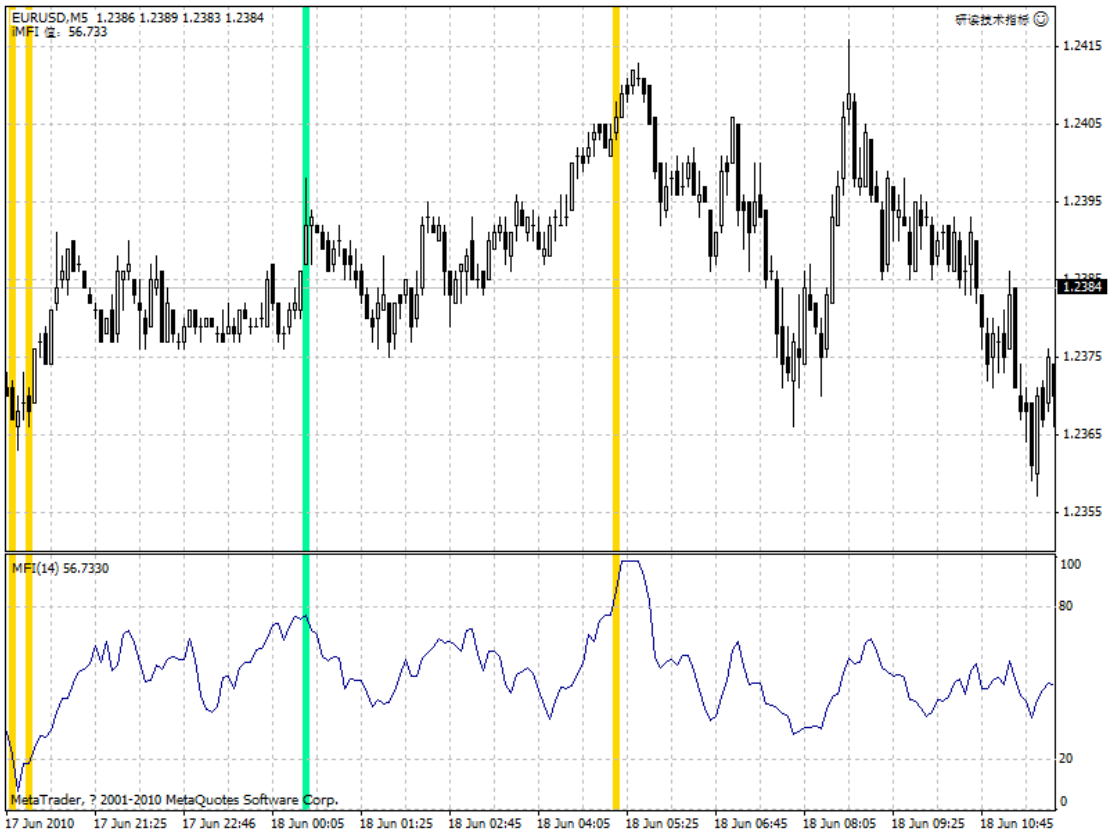
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 快速线周期,默认选 14 |
| 4、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 5、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iMomentum(NULL,0,14,PRICE_CLOSE,0)
```

# 4.20 Money Flow Index 资金流量指数指标

iMFI 属于成交量指标，用于判断市场的趋势。




【用法】

1、 MFI 指标在 20~80 之外，市场可能出现反转；

2、 MFI 是 RSI 的扩展，两者联合效果更好。

【语法】double iMFI( string symbol, int timeframe, int period, int shift)

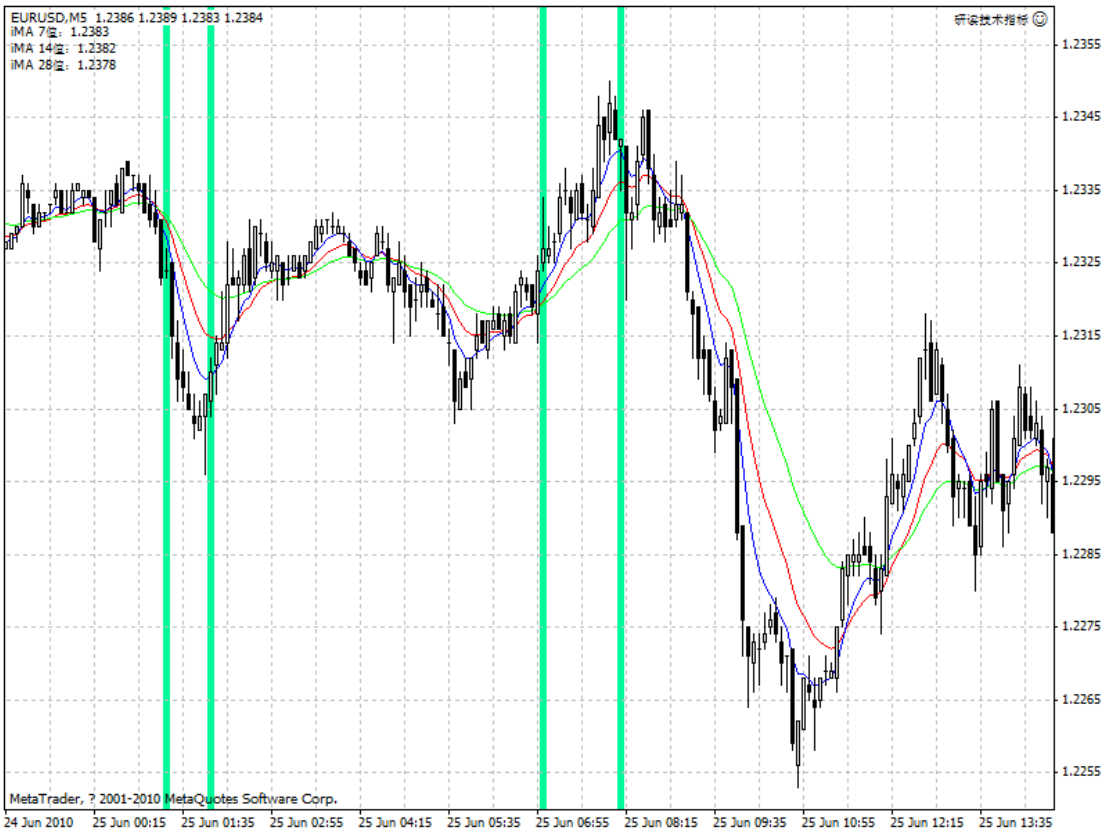
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 平均周期，默认选 14 |
| 4、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iMFI(NULL,0,14,0)
```

## 4.21 Moving Average 移动平均线指标

iMA 属于趋势指标，通常以 3 个不同周期的线组成一个指标体系。




【用法】

1、 价格小于 28、14 线时，跌势可能形成，可做空；

2、 价格小于 28、14、7 线时，保持跌势，可做空或继续持有空单；

3、 价格运行至 7、14 之间，空单平仓，观望；

4、 反之亦然。

5、 不同的货币对、不同的时间周期，参数设置不同。

【语法】double iMA( string symbol, int timeframe, int period, int ma_shift, int ma_method, int applied_price, int shift)

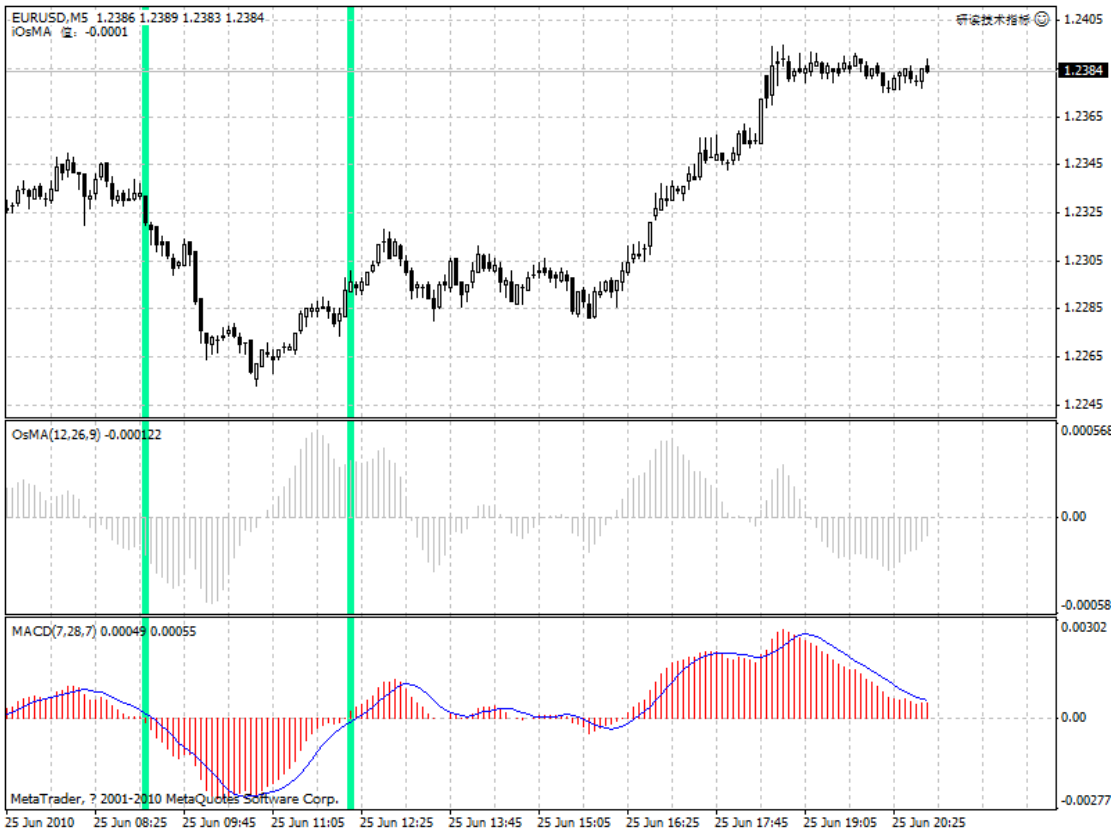
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 平均线周期，通常选 7、14、28 |
| 4、 | ma_shift | 偏移量，默认选 0 |
| 5、 | ma_method | MA 方法，通常选 MODE_EMA |
| 6、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 7、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iMA(NULL,0,7,0,MODE_EMA,PRICE_CLOSE,0)
iMA(NULL,0,14,0,MODE_EMA,PRICE_CLOSE,0)
iMA(NULL,0,28,0,MODE_EMA,PRICE_CLOSE,0)
```

## 4.22 Moving Average of Osillator 移动平均震荡指标

iOsMA 属于震荡指标，用于判断 CDMA 是否加速。




【用法】

【语法】double iOsMA( string symbol, int timeframe, int fast_ema_period, int slow_ema_period, int signal_period, int applied_price, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

3、 fast_ema_period 快速线周期，通常选 12

4、 slow_ema_period 慢速线周期，通常选 26

5、 signal_period 信号线周期，通常选 9

6、 applied_price 应用价格。默认选收盘价 PRICE_CLOSE

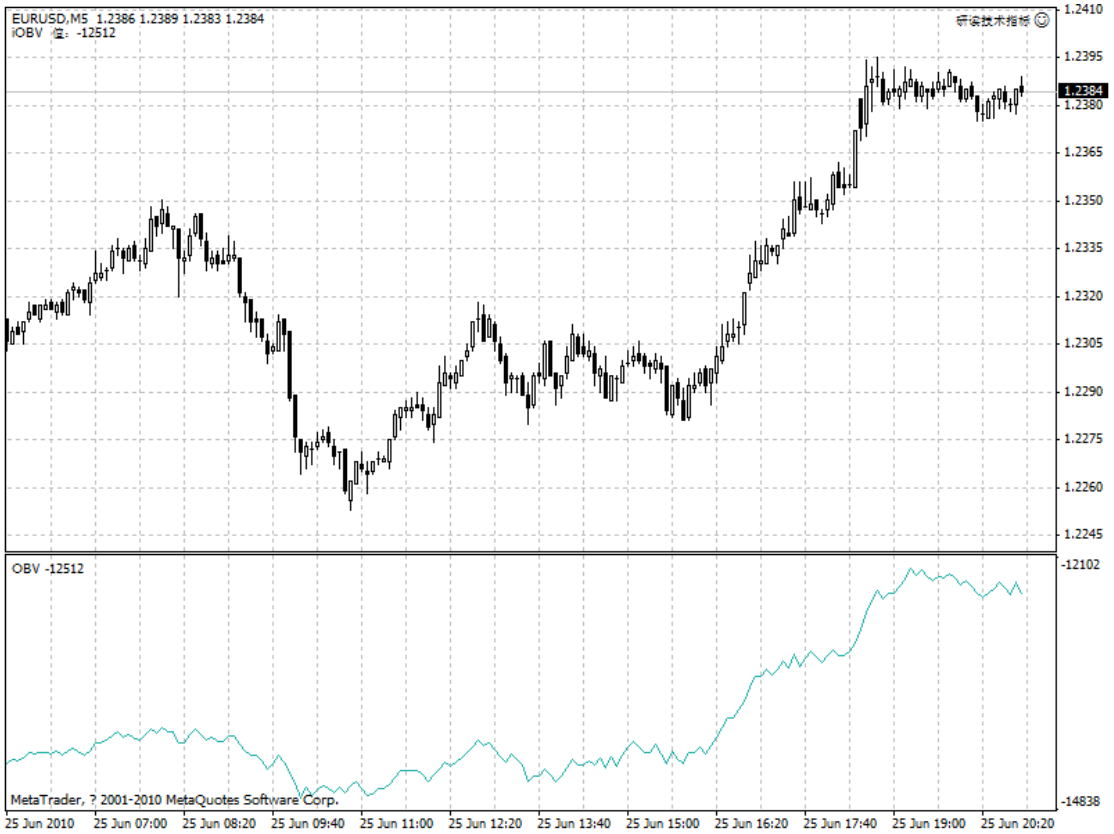
7、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

```
iOsMA(NULL,0,12,26,9,PRICE_CLOSE,0)
```

## 4.23 On Balance Volume 能量潮指标

iOBV 属于成交量指标，成交量和价格相互关联，给出市场趋势信号。




【用法】

需要与牛力指标、熊力指标联动。

【语法】double iOBV( string symbol, int timeframe, int applied_price, int shift)

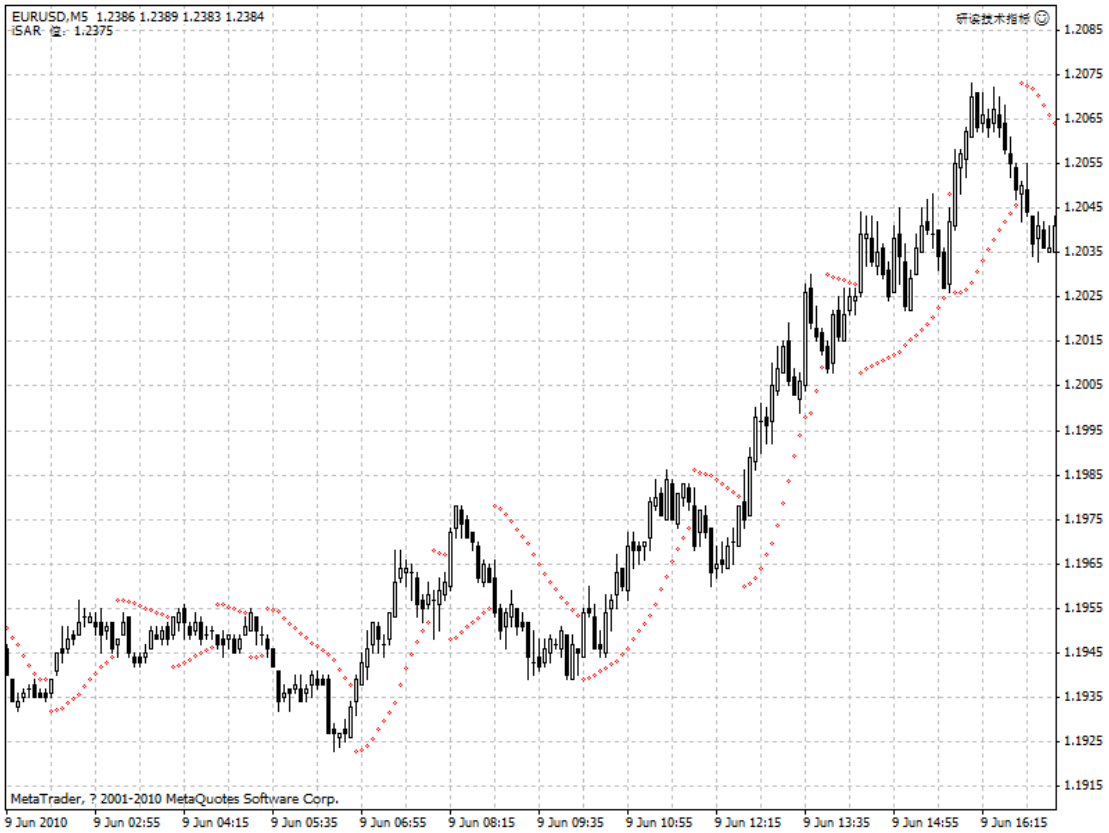
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | applied_price | 应用价格。默认选收盘价 PRICE_CLOSE |
| 4、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iOBV(NULL, 0, PRICE_CLOSE, 0)
```

## 4.24 Parabolic SAR 抛物线状止损和反转指标

iSAR 属于趋势指标，给出一个市场趋势结束或者开始信号。




【用法】

1、 SAR 读值低于价格水平，市场处于涨势，反之处于跌势；

2、 该指标过于敏感，需要其他指标配合。

【语法】double iSAR( string symbol, int timeframe, double step, double maximum, int shift)

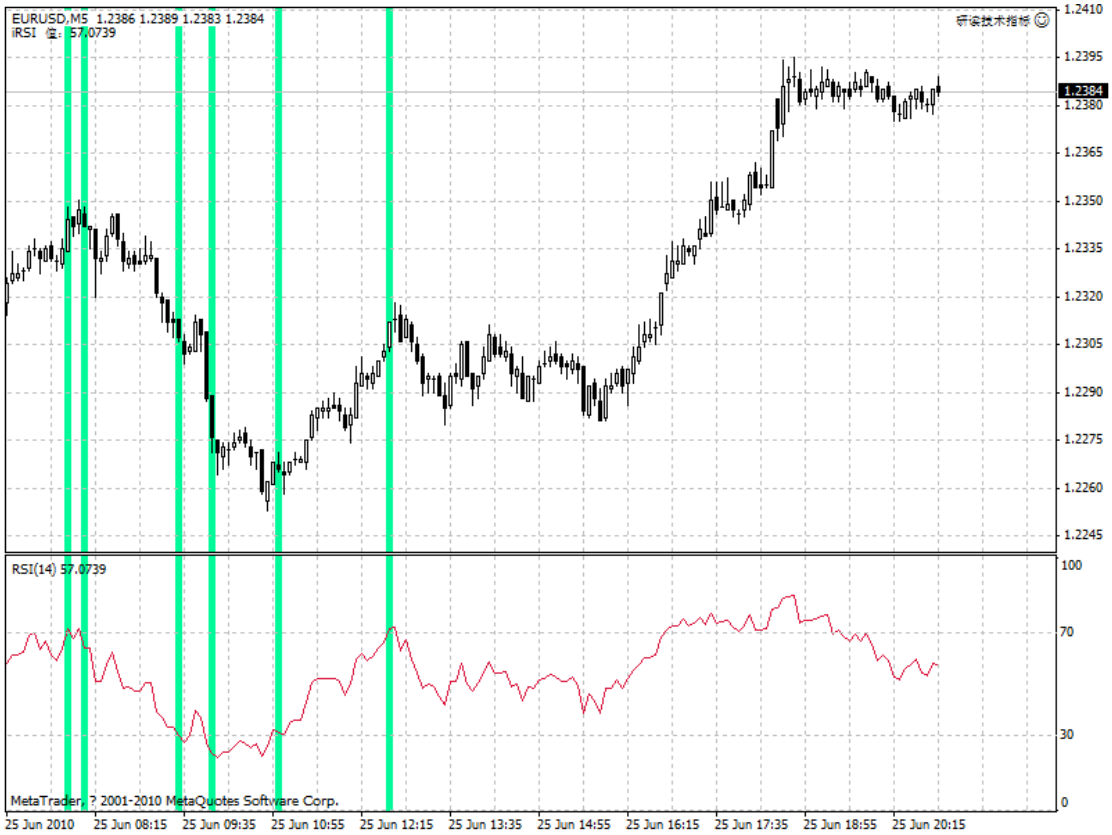
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | step | 步长，通常 0.02 |
| 4、 | maximum | 最大值，通常 0.2 |
| 5、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iSAR(NULL,0,0.02,0.2,0)
```

## 4.25 Relative Strength Index 相对强弱指标

iRSI 属于震荡指标，反映市场买卖强度。




【用法】

1、 RSI 读值超过 70，市场处于超买阶段，低于 30 处于超卖阶段；

2、 RSI 处于 30~70 之间，市场按照 RSI 方向发展。

3、 同时使用两个或三个不同周期的 RSI 曲线判断市场反转信号也是一种常见的做法。

【语法】double iRSI( string symbol, int timeframe, int period, int applied_price, int shift)

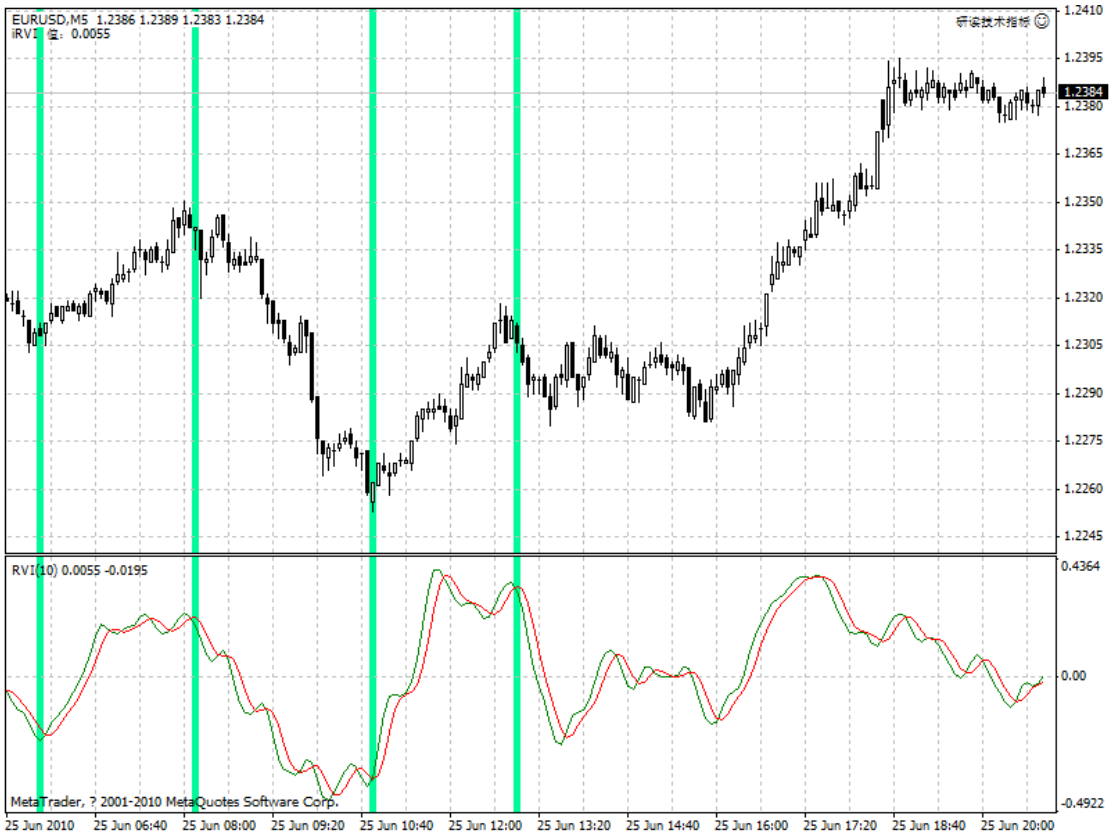
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 平均周期，通常 14 |
| 4、 | applied_price | 应用价格，通常 PRICE_CLOSE |
| 5、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iRSI(NULL,0,14,PRICE_CLOSE,0)
```

## 4.26 Relative Vigor Index 相对活力指数指标

iRVI 属于震荡指标，发出买卖信号。




【用法】

RVI 两线相交，发出买入卖出信号。

【语法】double iRVI( string symbol, int timeframe, int period, int mode, int shift)

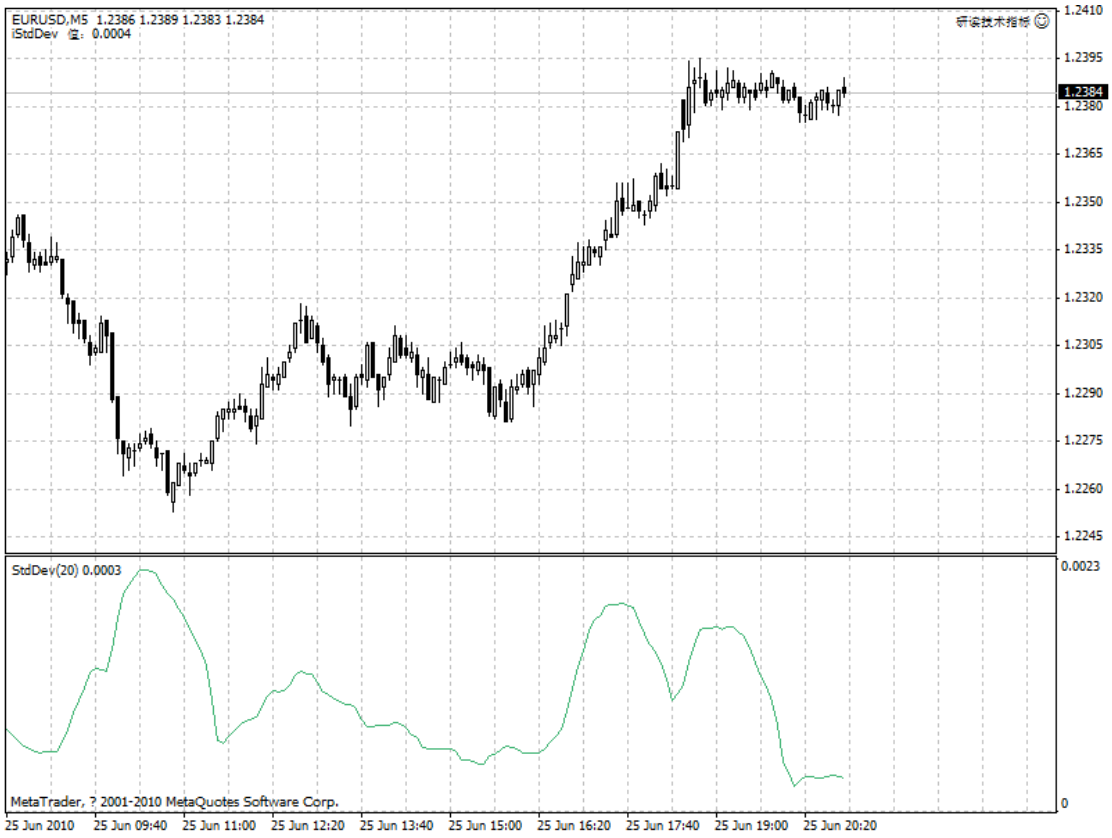
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 平均周期，通常 10 |
| 4、 | mode | 指标类型，MODE_MAIN，MODE_SIGNAL |
| 5、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iRVI(NULL, 0, 10,MODE_MAIN,0)
```

## 4.27 Standard Deviation 标准离差指标

iStdDev 属于趋势指标，反映市场活跃程度。




【用法】

StdDev 读数低，说明市场不活跃，读数高说明市场活跃。

【语法】double iStdDev( string symbol, int timeframe, int ma_period, int ma_shift, int ma_method, int applied_price, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | ma_period | 平均周期，通常 20 |
| 4、 | ma_shift | MA 偏移，通常 0 |
| 5、 | ma_method | MA 方法，通常 MODE_EMA |
| 6、 | applied_price | 应用价格，通常 PRICE_CLOSE |
| 7、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

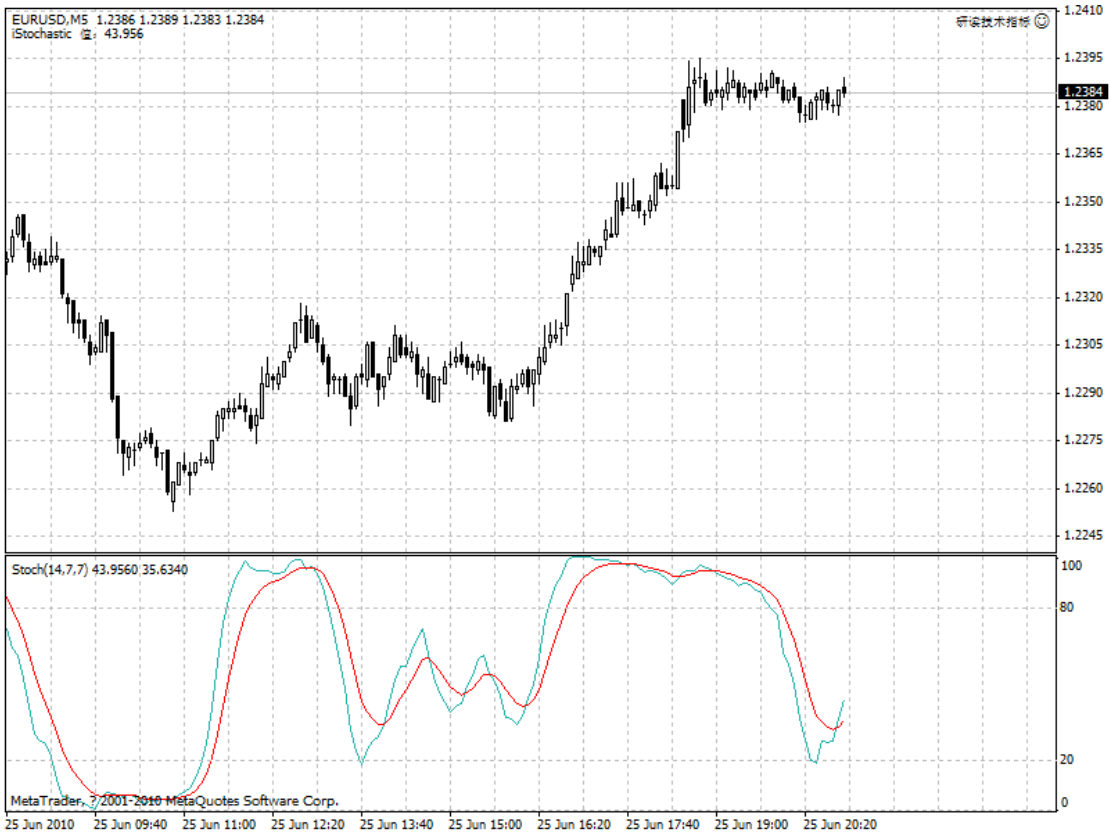
【代码】

```
iStdDev(NULL,0,20,0,MODE_EMA,PRICE_CLOSE,0)
```

## 4.28 Stochastic Oscillator 随机震荡指标

iStoch 属于震荡指标，又叫 KD 指标，提供买卖信号。

【用法】




1、 可使用神奇数字做检测；

2、 两线低于 20，再回升到 20 以上，做多；

3、 两线高于 80，再回落到 80 以内，做空；

4、 %K 高于%D 线，做多；

5、 %K 底于%D 线，做空。

【语法】double iStochastic( string symbol, int timeframe, int %Kperiod, int %Dperiod, int slowing, int method, int price_field, int mode, int shift)

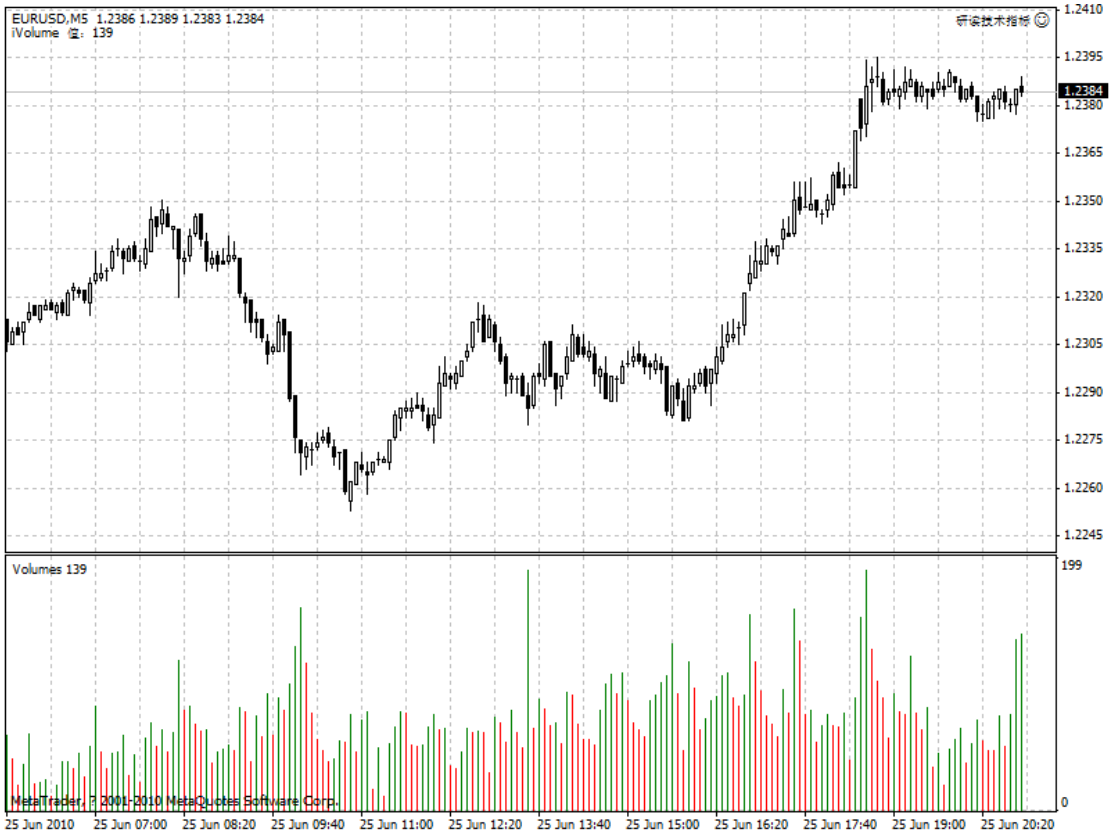
| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | %Kperiod | %K 周期，通常 14 |
| 4、 | %Dperiod | %D 周期，通常 7 |
| 5、 | slowing | 滚动值，通常 7 |
| 6、 | method | MA 方法 通常 MODE_EMA |
| 7、 | price_field | 价格参量，可以是以下值: 0 - Low/High 或者 1 - Close/Close |
| 8、 | mode | 指标类型，MODE_MAIN，MODE_SIGNAL |
| 9、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iStochastic(NULL,0,14,7,7,MODE_EMA,1,MODE_MAIN,0)
```

# 4.29 Volumes 成交量指标

iVolumes 在图表中显示成交量柱线。




【用法】

【语法】double iVolume( string symbol, int timeframe, int shift)

1、 symbol 指定货币对，NULL 为默认当前货币对

2、 timeframe 时间周期，0 为当前时间周期

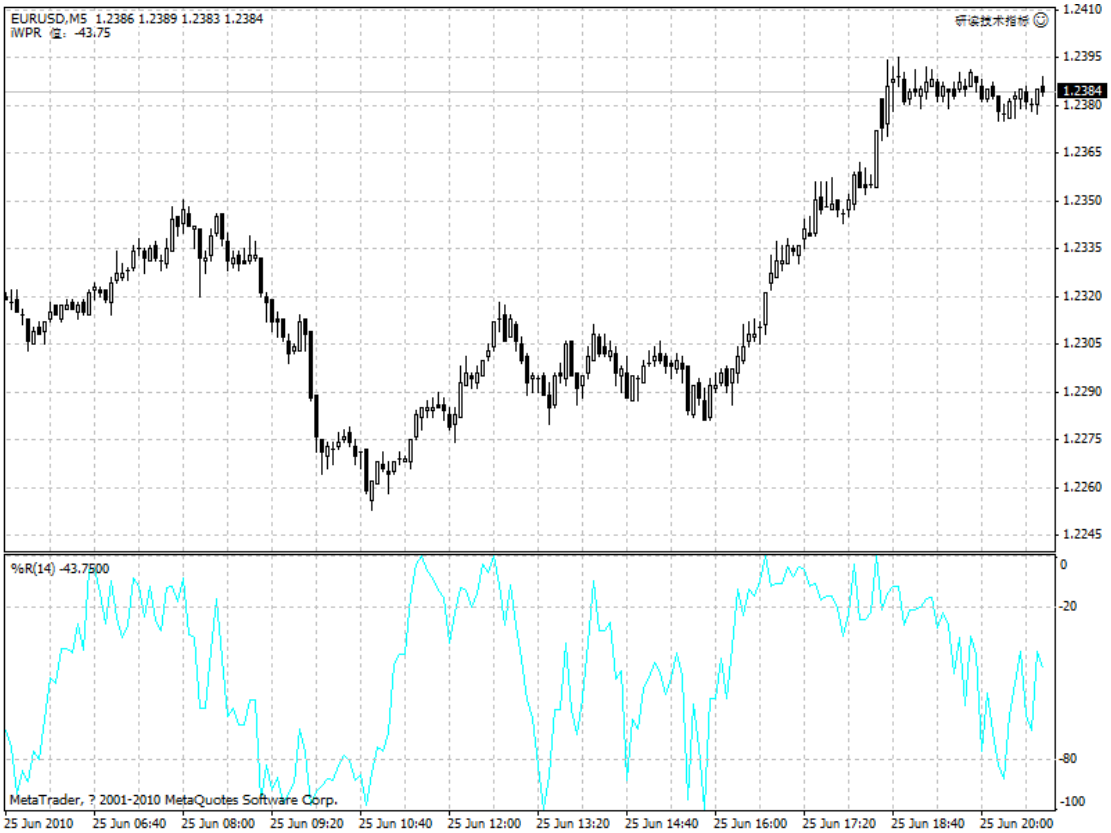
3、 shift 指定柱值，0 为当前柱，1 为前一个柱，以此类推

【代码】

iVolume(NULL,0,0)

# 4.30 Williams'Percent Range 威廉指标

iWPR 属于震荡指标，提示市场是否超买超卖。




【用法】

1、 WPR 能够大胆预测市场的反转；

2、 WPR 在 0~-20%之间，市场处于超买状态；

3、 WPR 在-80%~-100%之间，市场处于超卖状态。

【语法】double iWPR( string symbol, int timeframe, int period, int shift)

| | | |
|---|---|---|
| 1、 | symbol | 指定货币对，NULL 为默认当前货币对 |
| 2、 | timeframe | 时间周期，0 为当前时间周期 |
| 3、 | period | 平均周期，通常 14 |
| 4、 | shift | 指定柱值，0 为当前柱，1 为前一个柱，以此类推 |

【代码】

```
iWPR(NULL,0,14,0)
```